Desempenho e modelagem de SGBD Objeto-Relacional em sistemas de visualização de faturamento na WEB

Rogério Prado Colferai

Trabalho Final de Mestrado Profissional

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Colferai, Rogério Prado

C68d Desempenho e modelagem de SGBD Objeto–Relacional em sistemas de visualização de faturamento na WEB / Rogério Prado Colferai -- Campinas, [S.P. :s.n.], 2004.

Orientador : Geovane Cayres Magalhães.

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. Banco de dados relacionais. 2. Banco de dados orientado a objetos. 3. Modelagem de dados. I. Magalhães, Geovane Cayres. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Instituto de Computação
Universidade Estadual de Campinas

Desempenho e modelagem de SGBD Objeto-Relacional em sistemas de visualização de faturamento na WEB

Rogério Prado Colferai

Fevereiro de 2004

Banca Examinadora:

- Prof. Dr. Geovane C. Magalhães
  Instituto de Computação – UNICAMP (Orientador)
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação – UNICAMP
- Prof. Dr. Anderson Delcio Parreira
  CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
- Prof. Dr. Mario Lucio Cortes
  Instituto de Computação – UNICAMP (Suplente)

Desempenho e modelagem de SGBD Objeto-Relacional em sistemas de visualização de faturamento na WEB

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Rogério Prado Colferai

Campinas, 20 de fevereiro de 2004

Prof. Dr. Geovane Gayres Magalhães
Instituto de Computação – Universidade Estadual de Campinas (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na Área de Engenharia de Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 20 de fevereiro de 2004, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Anderson Delcio Parreira
CPqD - UNIP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC - UNICAMP

Prof. Dr. Géovane Cayres Magalhães
IC-UNICAMP

Resumo

Esta dissertação apresenta conceitos do modelo Objeto-Relacional e uma aplicação prática dos recursos desta nova tecnologia em um módulo visualizador de faturas via WEB, que hoje é suportado por um esquema de dados relacional. Foram aplicados neste trabalho novos recursos de modelagem, armazenamento de dados e potencialidades da Orientação a Objeto que são oferecidos por este novo paradigma visando assim, obter um esquema objeto-relacional que apresente ganho de desempenho na recuperação e atualização dos dados em relação ao original. Foram utilizados recursos de modelagem da UML (*Unified Modeling Language*) bastante populares no desenvolvimento de sistemas de informações no paradigma Orientado a Objetos. Um conjunto de consultas e atualizações foi submetido a uma base de dados real carregada no esquema relacional, da aplicação corrente, e objeto-relacional. O comportamento dos esquemas na execução deste conjunto de consultas e atualizações foi analisado nos aspectos de desempenho e complexidade.

Abstract

This dissertation presents concepts of the Relational-Object model and a practical implementation of this new technology's resources in a billing viewer module via Web, which nowadays is supported by a relational data schema. In this work we applied new resources of data storage and Object-Oriented power that are offered by this new paradigm, aiming to obtain a relational-object schema that presents performance gain on data recovering and updating, when compared to the original one. UML (Unified Modeling Language) modeling resources very popular in Object Oriented paradigm Information Systems development were used. A sort of selects and updates were submitted to a real database loaded in the relational schema, of the current application, and in the relational-object schema. The behavior of both schemas in this sort of selects and updates was analyzed in performance and complexity aspects.

Agradecimentos

Primeiramente gostaria de agradecer a minha mãe, Maria Helena pela compreensão nas ausências, pela força nas dificuldades e pelo carinho em toda caminhada.

Em memória de meu pai Pedro, que sempre me incentivou aos estudos e deixou um modelo exemplar de vida para minha família.

Agradeço também aos meus irmãos, Raphael e Renan e a todos meus familiares que, tenho certeza, também torceram por mim.

A minha namorada Monise, pela compreensão e paciência destes anos todos.

Aos colegas Sindo, Márcia, Carlinhos, Rita, Paula, Andreza e Flávio, pelas inúmeras coisas que fizeram por mim e para mim. Obrigado pelas sugestões que ajudaram a resolver problemas encontrados, pelos vários momentos de alegria que fizeram amenizar os obstáculos. Enfim obrigado pela família que me proporcionam.

Agradeço também aos inúmeros amigos que conquistei durante este tempo, e aos que já me acompanham, obrigado pelos ótimos momentos vividos. Em especial, ao grupo de trabalhos pelas oportunidades, sugestões e críticas.

Ao meu orientador Geovane, obrigado pela credibilidade e ensinamentos. Que outros também possam ver que além de bom profissional também se é possível ser ótimo ser humano.

A Deus pela determinação e força para sempre prosseguir, independente do que fosse necessário superar.

Obrigado a todos que direta ou indiretamente me ajudaram na realização deste trabalho.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução e Motivação

Uma característica fundamental da abordagem de banco de dados é fornecer diversos níveis de abstração dos dados, possibilitando por exemplo, a omissão de detalhes de armazenamento que são desnecessários para os usuários [KYTE8I]. Os modelos de dados são algumas das ferramentas que permitem esta abstração. Estes modelos são baseados em diversos conceitos fornecidos para descrever a estrutura do banco de dados. Os modelos conceituais são utilizados para representar os dados da forma como eles são vistos pelos usuários[INFO98]. Os modelos físicos fazem uma descrição em função de um sistema de gerência de banco de dados específico. Entre estes dois extremos estão os modelos lógicos, que escondem alguns detalhes de armazenamento dos dados, mas podem ser implementados de uma forma mais direta que os modelos conceituais[EN94].

Atualmente, o modelo de dados lógico mais utilizado por SGBDs (Sistema de Gerenciamento de Banco de Dados) comerciais é o Relacional, que é baseado em extensões matemáticas do conceito de relações[DATE91]. Estas relações representam entidades ou relacionamentos do mundo real e possuem atributos de tipos fixos que descrevem as suas propriedades. Entretanto, os fabricantes alegam que os SGBDs Relacionais não atendem as demandas das novas tecnologias, como projetos de engenharia, imagens, bancos de dados científicos, sistemas de informações geográficas, multimídia, entre outros objetos complexos encontrados no mundo real. Estas aplicações possuem requisitos e características que diferem do armazenamento e recuperação de dados tradicionais, incluindo estruturas para objetos, novos tipos de dados e grandes itens textuais[WILH99].

Para suprir as limitações dos SGBDs Relacionais nos últimos anos, novos SGBDs foram desenvolvidos. Entre os novos sistemas, além dos SGBDs Orientados a Objetos, surgiram os SGBDs Objeto-Relacional[ORA9I], que serão foco de nossos estudos neste trabalho.

Os SGBDs baseados no modelo Orientado a Objetos foram propostos para atender às necessidades das novas aplicações, incluindo objetos com estruturas mais complexas, novos tipos de dados e a definição de operações específicas das aplicações. O modelo Orientado a Objetos

oferece a flexibilidade de manipular estes requisitos, sem ser limitado pelos tipos de dados e linguagens de consultas disponíveis nos sistemas de banco de dados tradicionais [DORN]. Com estas vantagens, muitas pessoas acharam que seria o fim do modelo Relacional, que haveria uma nova revolução já que no paradigma Orientado a Objetos se tinha uma maneira muito mais natural para se modelar qualquer objeto do mundo real. Objetos complexos que antes eram um verdadeiro tormento para os analistas seriam tratados como quaisquer outros objetos, com seus atributos e métodos próprios. Essa impressão começou a ser modificada quando se viu que esses bancos de dados, por serem totalmente diferentes dos bancos Relacionais, não estavam suficientemente maduros em algumas características fundamentais que os clientes exigiam como robustez, confiabilidade, performance, capacidade de armazenamento e facilidade de consultas entre outras, que os bancos Relacionais ofereciam. Isto se verificava porque os bancos Relacionais tinham anos de desenvolvimento e aperfeiçoamento constante, e todo esse *know how*, tinha que ser jogado fora para se começar tudo de novo com os bancos Orientados a Objetos[ARRU00].

Nesse contexto é que nasce o paradigma Objeto-Relacional, que visa ser mais representativo em semânticas e construções de modelagens do que os SGBDs Relacionais. Estes SGBDs se aproveitam da versatilidade, robustez e tecnologia consagradas dos SGBDs Relacionais, incorporando a flexibilidade e potencialidade da Orientação a Objetos em um único produto, permitindo a recuperação de objetos complexos através da extensão do conteúdo Relacional[ARRU00].

Outra vantagem deste modelo é que continua oferecendo interface Relacional para as aplicações desenvolvidas neste paradigma, dispensando assim a necessidade de dois bancos de dados, um para as aplicações Relacionais e outro para aplicações Orientadas a Objetos[ORA9I].

Entretanto, não se pode afirmar que existe um modelo lógico padronizado que atenda a todos os SGBDs Objeto-Relacional disponíveis no mercado. Diferentemente dos SGBDs Relacionais, que são sustentados por um modelo formal como ferramenta de modelagem, os SGBDs Objeto-Relacional surgiram das diversas implementações dos produtos comerciais.

Sendo assim, em função desta nova tecnologia, surge a necessidade de um processo para mapear um esquema Relacional existente em um Objeto-Relacional visando disponibilizar as potencialidades da Orientação a Objetos, o que já é um grande passo do ponto de vista das tendências de mercado.

Neste trabalho apresentaremos as principais características de implementação do modelo Objeto-Relacional através de exemplos para facilitar a compreensão, e uma aplicação prática destas potencialidades em um módulo visualizador de contas On-Line via WEB para sistemas de Faturamento, que hoje é um protótipo suportado por um modelo de dados Relacional.

Com o incremento das potencialidades providas por este novo paradigma, visamos um considerável ganho de performance na recuperação e atualização dos dados, e, ainda, prover maior naturalidade no desenvolvimento e compreensão das estruturas através do uso da UML (*Unified Modeling Language*), uma linguagem de modelagem rica, padronizada e bastante popular no desenvolvimento de sistemas de informações no paradigma Orientado a Objetos.

Para demonstrar um comparativo real de desempenho entre o modelo Relacional, que hoje suporta a aplicação, e o Objeto-Relacional, que criaremos neste trabalho, apresentaremos um capítulo que descreverá o comportamento de cada modelo depois de submetidos a várias análises comparativas com a mesma massa de dados.

Esta dissertação foi desenvolvida na empresa CPqD - Centro de Pesquisa e Desenvolvimento - no departamento de Soluções em Billing (Faturamento) do CPqD - e subsidiada pelo FUNTELL (Fundo para o Desenvolvimento Tecnológico das Telecomunicações).

O FUNTELL tem como foco a inovação tecnológica em telecomunicações, acesso a recursos de capital para pequenas e médias empresas de base tecnológica no setor de telecomunicações, capacitação de recursos humanos em tecnologia e pesquisa aplicada às telecomunicações.

Capítulo 2

Conceitos Básicos

Este capítulo apresenta alguns conceitos básicos necessários ao entendimento da dissertação. O capítulo está organizado da seguinte forma. A seção 2.1 nos mostra os processos básicos de Sistemas de Faturamento aplicados em Telecom, que na realidade nos servirá de base para compreensão de algumas informações tratadas neste trabalho. A seção 2.2 apresenta um protótipo de um módulo de visualização de contas chamado EBV (*Electronic Billing Viewer*), que apresenta informações de faturas disponibilizadas pelo sistema de Faturamento em documentos On-Line via WEB. As seções 2.3 e 2.4 nos mostram conceitos básicos dos modelos: Relacional e Orientado a Objetos, que servirão de base para a compreensão do modelo Objeto-Relacional, que é o foco de nossos estudos e será detalhado no próximo Capítulo. A seção 2.5 apresenta um resumo do Capítulo.

2.1 Sistema de Faturamento

Um sistema completo de Faturamento aplicado em Telecom tem como principais características garantir a execução do processo desde a entrada de CDRs (*Call Detail Record*), que são registros contendo informações das chamadas telefônicas, até a emissão da conta incluindo atendimento a reclamação de contas.

A Figura 1 nos mostra os módulos funcionais básicos a partir da entrada dos arquivos de CDRs.

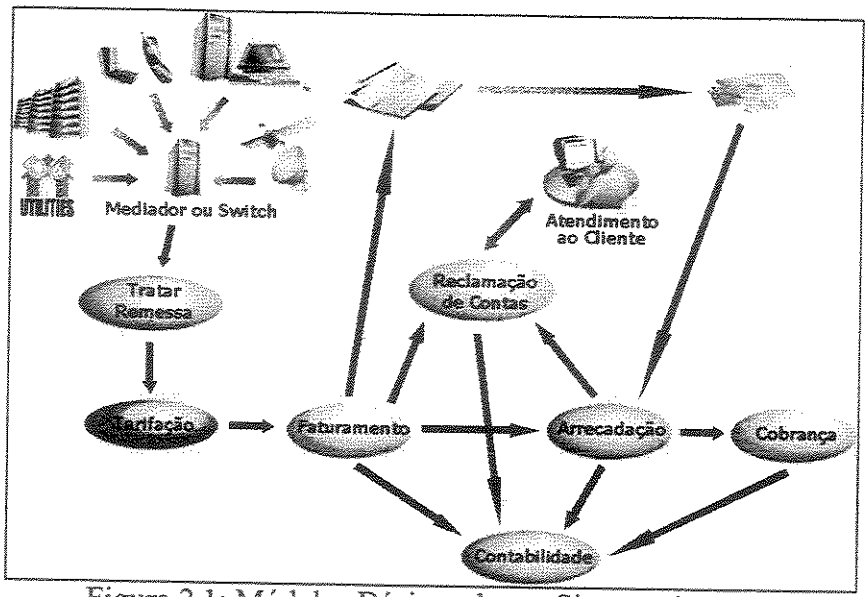

Figura 2.1: Módulos Básicos de um Sistema de Faturamento.

O módulo Tratar Remessa é responsável pelo processamento de todas as entradas de dados do sistema. Funcionalmente é subdivido em Carga, Crítica, Valoração e Apropriação. Cada um desses módulos desempenha uma função específica na seqüência do processamento das informações recebidas nos arquivos de CDRs.

A Figura 2 descreve a seqüência de processos pelos quais os CDRs recebidos são submetidos.

Figura 2.2: Seqüência de processos no tratamento das entradas

Os CDRs gerados nas várias origens (bilhetadores próprios, remessas DACC (*Debito Automático em Conta Corrente*), terceiros, etc...) são, inicialmente, submetidos a um pré-processador, e posteriormente submetidos ao processo de carga.

Baseado em parametrizações efetuadas na configuração do sistema, e em leiautes previamente definidos, os arquivos são abertos e analisados. O escopo dessa análise é puramente sintática, visando garantir a integridade estrutural dos dados recebidos. Em outras palavras, o processo de carga garante que uma remessa recebida é identificada como validada pelo sistema e que seus dados sejam carregados na base.

Em seguida o processo de análise utiliza as informações armazenadas em banco de dados pelo processo de Carga e efetua a crítica semântica, ou seja, validações com o cadastro da empresa sobre o conteúdo dos CDRs. São analisadas e extraídas as seguintes informações:

- A origem dos assinantes A e B
- As localidades a que pertencem
- A distância entre as localidades
- O horário da chamada
- A duração da chamada
- Qual o serviço envolvido
- O tipo da chamada (Nacional ou Internacional)
- A direção de cobrança (A ou B)

Com a finalização da análise, onde todas as informações necessárias já foram extraídas dos CDRs, segue-se com o processo de valoração. Este processo leva em conta todas as regras de negócio parametrizadas pela empresa, inclusive o plano específico associado a cada terminal. Estes dados são valorados e darão origem a itens de faturamento, que são chamadas ou serviços prestados ao cliente, que irão compor a fatura propriamente dita, onde cada item desta fatura será relacionado ao assinante responsável pelo processo de Apropriação.

Em seguida o faturamento recebe como entrada, os itens de chamadas gerados pela Tarifação, devidamente valorados e apropriados. Recebem também itens gerados a partir das leituras de serviços medidos, itens provenientes de outras entradas, tais como reclamação de contas, arrecadação e cobrança, e os itens relativos às remessas de cobrança de serviços de

terceiros, que serão cobrados em conta telefônica. A saída do faturamento é um arquivo contendo todos os dados necessários à impressão da conta.

A partir deste momento o módulo de visualização de contas EBV será capaz de prover aos clientes da empresa operadora, consultas sobre suas contas através da WEB. Este tópico será detalhado na próxima seção.

Prosseguindo com as atividades do Sistema de Faturamento após o faturamento, teremos a arrecadação, que é responsável pela consolidação do pagamento de documentos emitidos pelo Faturamento com os devidos encargos, recebendo estas informações dos agentes arrecadadores. Funcionalmente está dividido em: carga das informações de pagamento, crítica, baixa e fechamento das remessas, agentes arrecadadores e movimento de arrecadação.

No final deste processo têm-se os documentos pagos, os devidos encargos calculados e ainda o controle de valores movimentados entre os agentes arrecadadores e a empresa operadora.

A cobrança é o próximo passo, e se responsabiliza pelo tratamento de contas e documentos de recebimento que estejam em situações pendentes. Aplica funcionalidades como sanções de cobrança, ações de posto de cobrança, alteração da situação de cobrança e cancelamento por débitos incobráveis.

Atendimento a clientes é um módulo responsável pelo tratamento das solicitações dos clientes relacionados às contas emitidas pelo Sistema de Faturamento. É divido em três blocos funcionais: o primeiro chamado de *Reclamação de Contas*, onde o cliente pode discutir itens relacionados a uma conta vencida ou a vencer; o segundo disponibiliza as várias facilidades em relação à conta, e o terceiro bloco utiliza as informações geradas a partir do primeiro bloco, para possibilitar aos especialistas a apuração das reclamações.

Como resultado final do processo de Reclamação de Contas tem-se a geração dos Bilhetes de Reclamação, que são encaminhados para o processo de Apuração. Têm-se também as informações de crédito/débito aos clientes, que são encaminhadas para o Departamento Financeiro da empresa.

Com as etapas mencionadas acima, cobrimos toda a cadeia básica de processos que um sistema de Faturamento aplica no setor de Telecom.

2.2 Módulo de visualização de contas EBV

A privatização e globalização estão mudando a indústria de telecomunicações ao redor do mundo. No entanto, a grande mudança está sendo guiada pelo crescimento explosivo da Internet, uma verdadeira revolução.

Com o objetivo de baixar custos e mostrar modernidade, as empresas em geral já vêm disponibilizando diversos serviços via WEB, entre eles, a apresentação e pagamento de contas.

O módulo EBV (Electronic Billing Viewer) é um protótipo de um visualizador On-line de contas via WEB, que recebe como entrada dados processados por um sistema de Faturamento, e proporciona aos clientes da Empresa de Telecomunicações a facilidade de visualizar com antecedência a conta a ser paga, podendo imprimi-la, caso deseje. Para que os clientes tenham acesso ao sistema, é necessário o cadastramento do mesmo para fins de segurança das informações.

O diagrama abaixo demonstra o recebimento das informações do Sistema de Faturamento, bem como o acesso dos clientes ao módulo EBV.

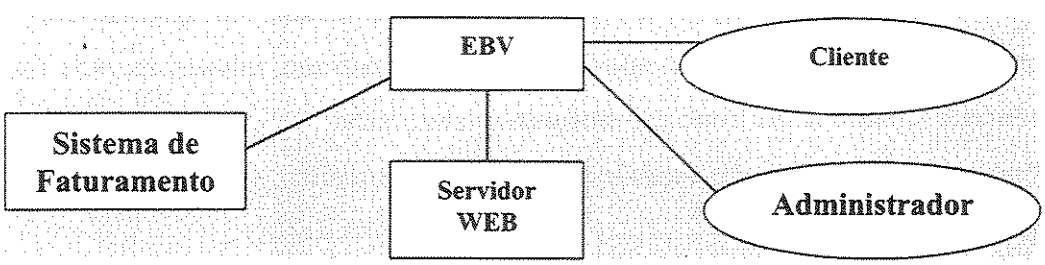

Figura 2.3: Visão geral do EBV

Atualmente, o protótipo disponibiliza as seguintes visualizações:

- **Visualização Histórica de Contas**
- **Visualização Conta Resumida**
- **Visualização Conta Detalhada**
- **Visualização Sumarização Livre**

Visualização Histórica de Contas

A visualização do histórico de contas apresenta os últimos documentos de recebimento do cliente responsável pelo número do telefone informado. Esta visualização é mostrada assim que o usuário efetua o *login* no sistema e apresenta o valor total das últimas contas do número do telefone informado.

Para cada documento de recebimento, são mostradas as seguintes informações:

- Número do telefone responsável pelo documento;
- Período de referência (data início e data fim);
- Valor total do documento;
- Saldo do cliente, mostrado apenas quando a empresa operadora parametriza o tipo de baixa a ser realizada por saldo, e não baixa por documento de recebimento.

Visualização Conta Resumida

A visualização resumida apresenta os dados sumarizados do documento de recebimento selecionado, podendo ser a sumarização de um telefone, do documento ou do documento inteiro.

No início da página de visualização de Conta Resumida, são apresentadas informações como: nome do cliente, número do responsável pelo documento, número da nota fiscal, período de referência e número do telefone que está sendo consultado.

Para cada documento de recebimento, os valores devem ser sumarizados por bloco de serviço.

Se o bloco de serviço for do tipo chamada, como por exemplo, Ligações Internacionais, o total de minutos e os valores referentes serão sumarizados também por grupo horário, ou seja, pelo tipo da tarifa que foi cobrada (reduzida, normal).

No final da página de visualização, são obtidos os valores totais de chamadas e o total geral.

Visualização Conta Detalhada

A visualização detalhada apresenta os dados de um número de telefone de forma detalhada, a partir da seleção de um documento de recebimento.

No início da página de visualização de Conta Detalhada são apresentadas informações como: nome do cliente, número do responsável pelo documento, número da nota fiscal, período de referência e também o número do telefone que está sendo consultado.

Para cada documento de recebimento, os valores devem ser detalhados por Bloco de Serviço.

Se o bloco de serviço for do tipo chamada, como por exemplo, Ligações Internacionais, é possível visualizar detalhes da chamada como Data, Telefone e Localidade de Destino, Hora início da ligação, Duração, Grupo Horário, ou seja, pelo tipo da tarifa que foi cobrada (Reduzida, Normal), e o valor cobrado pela chamada.

Se o bloco de serviço for diferente do tipo chamada, por exemplo, Desconto Nacional, podem-se visualizar informações como descrição e valor.

No final da página de visualização, são totalizados os valores das chamadas.

Visualização Sumarização Livre

A sumarização livre apresenta informações referentes às chamadas de um determinado documento de recebimento, agrupadas segundo critérios que o usuário seleciona.

O usuário deve selecionar no mínimo um e no máximo dois critérios para agrupar os dados referentes a chamadas do documento selecionado.

As informações estarão agrupadas de acordo com os critérios selecionados pelo usuário.

Os critérios disponíveis são:

- Tarifa: agrupar por grupo horário (normal, reduzida)
- Telefone origem: agrupa por telefone de origem da chamada
- Telefone destino: agrupa por telefone de destino da chamada
- Serviços de uso: agrupa por blocos de serviço (DDD, DDI)
- Localidade Destino: agrupa por localidade de destino da chamada

Os dados de chamada estarão ordenados de forma ascendente e agrupados de acordo com os critérios selecionados, devendo ser sumarizados as informações: total de chamadas, duração total e valor total.

2.3 Modelo Relacional

O modelo relacional foi formalmente definido no Laboratório da IBM em San Jose - Califórnia, em 1970. O projeto inicial foi denominado de Sistema R e definia a organização dos dados e linguagens formais para a sua manipulação. Com base nestas linguagens formais, a primeira versão da linguagem SQL (*Structured Query Language*) foi definida. Esta linguagem é, atualmente, um padrão para gerenciamento de dados em SGBDs relacionais[RH96].

O estudo do modelo de dados relacional apresenta 3 aspectos[FERN8I]:

- **Aspectos estruturais**: formalizam matematicamente a maneira como os dados estão organizados no modelo. Esta formalização é baseada na teoria dos conjuntos;
- **Aspectos de integridade**: descrevem os procedimentos para garantia de integridade de dados quando da ocorrência de operações de atualização de dados;
- **Aspectos de manipulação**: descrevem as linguagens formais e comerciais definidas para o modelo.

Com base nesta estrutura demonstrada acima apresentaremos alguns conceitos básicos do modelo relacional como: domínio, atributo, tupla, relação e chave.

Domínio

Define o universo de valores permitidos para um certo item de dado, como por exemplo:

Domínio(idade) = [0,150];

Domínio(sexo) = (masculino, feminino);

Um domínio compreende um tipo de dado predefinido, mais um conjunto de restrições de integridade que limitam os valores permitidos para este tipo de dado. Por exemplo, o item de dado 'idade', apresenta um domínio inteiro mais uma restrição de integridade que delimita o

intervalo de valores permitidos para [0,150]. Assim, os valores deste dado ficam coerentes com a realidade em questão.

Domínios podem ser simples, ou seja, possuem um único valor, como um inteiro, ou compostos, quando possuem vários valores, como uma data, que apresenta dia, mês e ano. Os valores dos itens de dados que apresentam domínios compostos são abstraídos e tratados como um único valor. Uma data poderia ser manipulada como um *string*, por exemplo.

A noção de domínio de um item de dado assemelha-se a noção de domínio de um conjunto na matemática.

Atributo

É um nome dado a um domínio, utilizado para representar um item de dado. Matematicamente, um atributo é o nome de um conjunto. Em uma tabela, representa uma coluna. Pode apresentar um valor condizente com o domínio associado a ele ou *null*. *Null* indica ausência de valor ou valor desconhecido.

Um atributo pode conter apenas um valor atômico, ou seja, um valor indivisível. Um contra-exemplo seria definir um atributo estruturado Endereço, que apresentasse 4 valores: rua, número, cidade e CEP. Este tipo de atributo não é suportado pelo modelo relacional.

Tupla

É um conjunto de pares (atributo e valor), que define uma linha da tabela, ou seja, uma ocorrência de uma entidade ou relacionamento. Se imaginarmos em uma tabela como sendo um conjunto, uma tupla seria um elemento deste conjunto.

Relação

Pode ser definida como um subconjunto do produto cartesiano dos domínios D1, D2, ..., Dn, que correspondem aos atributos A1, A2, ..., An de uma tabela. O produto cartesiano gera D1 X D2 X ... X Dn tuplas, e a relação mantém apenas aquelas tuplas que contêm valores de atributos relacionados que estão presentes na realidade. Por isto, uma relação é um subconjunto.

Uma relação é vista como uma tabela no modelo relacional, composta por um cabeçalho e um corpo. O cabeçalho é formado por um conjunto fixo de atributos que possuem nomes distintos, para evitar ambigüidade na localização de um item de dado. O grau de uma relação

significa o número de atributos da mesma. O corpo de uma relação é formado por um número variável de tuplas, onde a ordem das mesmas não é significativa, ou seja, dada uma relação R1 com 3 tuplas, nesta ordem: t1, t2 e t3; e uma relação R2, com 3 tuplas, nesta ordem: t2, t3 e t1; a igualdade R1 = R2 é verdadeira. A cardinalidade de uma relação significa o número de tuplas da mesma.

O conceito de relação é ligeiramente diferente do conceito de tabela. Relação equivale à noção de conjunto, ou seja, um agrupamento de elementos sem repetição. Tabela equivale à noção de coleção, ou seja, um agrupamento de elementos onde é permitida a repetição. SGBDs lidam, na prática, com o conceito de tabela[SK98].

Chave

É fundamental no modelo para garantir identificação de tuplas e estabelecer relacionamentos entre relações. O modelo relacional define 2 tipos de chave [MCFA99]:

- **Chave primária (*pk*)**: atributo ou grupo de atributos que permite a identificação única de uma tupla em uma relação, ou seja, o valor da *pk* de uma tupla nunca se repetirá nas demais tuplas da mesma relação. Uma *pk* é escolhida dentre várias chaves candidatas, que podem estar presentes na relação. As chaves candidatas não selecionadas são ditas chaves alternativas. A cada chave alternativa deve ser associada uma RI que garanta que seus valores sejam únicos (*unique*);

- **Chave estrangeira (*ek*)**: atributo ou grupo de atributos de uma relação R1 que estabelece um relacionamento de equivalência, por valor, com uma *pk* de uma relação R2. O domínio da *ek* de R1 deve ser compatível com o domínio da *pk* de R2. Nada impede que R1 e R2 sejam a mesma relação.

Os aspectos de integridade do modelo relacional estão associados aos conceitos de chave primária e chave estrangeira. Garantir a integridade de um esquema relacional significa garantir o acesso individualizado a todas as tuplas de uma relação, assim como garantir relacionamentos válidos e condizentes com a realidade. O modelo relacional é regido por 2 regras de integridade básicas: a regra de integridade de entidade e a regra de integridade referencial.

A regra de integridade de entidade diz respeito à chave primária de uma relação. Ela diz que nenhum atributo que faz parte da chave primária pode ter *null* em alguma tupla. A manutenção desta regra garante que toda tupla possa ser identificada unicamente.

A regra de integridade referencial diz respeito às chaves estrangeiras de uma relação. Ela diz que o valor de um atributo que faz parte de uma chave estrangeira pode assumir *null* desde que o mesmo não faça parte da chave primária. Ainda, este mesmo atributo pode assumir um valor qualquer, condizente com o seu domínio, desde que este mesmo valor exista na chave primária de uma tupla da relação referida por ele. Com isto, nunca existirão relacionamentos incorretos entre dados[SK98].

Para que estas regras sejam sempre respeitadas, o SGBD deve implementar rotinas de verificação automática. Isto implica em testes e ações a serem realizadas pelo SGBD todas as vezes que uma operação de atualização for submetida ao mesmo. No caso da regra de integridade de entidade, o seguinte algoritmo deve ser executado toda vez que for incluída uma nova tupla em uma relação ou for alterado o valor de um atributo de uma tupla que faz parte da chave primária de uma relação [MCFA99]:

SE a chave primária da tupla for null OU existir outra tupla com o mesmo valor de chave primária

ENTÃO Impedir a efetivação da operação

SENÃO Efetivar a operação;

Já para o caso da regra de integridade referencial, três alternativas são geralmente tomadas pelo SGBD quando se percebe que uma violação da mesma irá ocorrer:

- **Impedimento**: a operação não é efetivada;
- **Cascata**: para o caso de exclusões de tuplas ou alterações de chave primária na relação referida, realizar a mesma operação em todas as tuplas que se referem a ela;
- **Anulação**: para o caso de exclusões de tuplas ou alterações de chave primária na relação referida, altera-se para *null* o(s) atributo(s) que compõe(m) a chave estrangeira que estabelece o relacionamento com esta relação (caso 1). Uma

variante desta alternativa é anular apenas a chave estrangeira de uma única tupla (caso 2).

A aplicação destas ações depende da operação de atualização submetida e da relação que sofre a operação (a referida ou a que possui uma referência).

Se uma operação de atualização é submetida à relação que possui a referência (possui a chave estrangeira), o seguinte algoritmo deve ser executado toda vez que for incluída uma nova tupla ou for alterado o valor de um atributo que faz parte da chave estrangeira:

SE a chave estrangeira da tupla for null
ENTÃO
SE a chave estrangeira fizer parte da chave primária
ENTÃO Impedir a efetivação da operação
SENÃO Efetivar a operação
SENÃO SE não existir uma tupla na relação referida com valor de chave primária igual ao valor da chave estrangeira desta tupla
ENTÃO SE a chave estrangeira fizer parte da chave primária
ENTÃO Aplicar impedimento
SENÃO Aplicar impedimento OU Aplicar anulação (caso 2)
SENÃO Efetivar a operação;

Se uma operação de atualização é submetida à relação referida (possui a chave primária), o seguinte algoritmo deve ser executado toda vez que for excluída uma tupla ou for alterado o valor de um atributo que faz parte da chave primária:

SE a chave estrangeira fizer parte da chave primária
ENTÃO Aplicar impedimento OU Aplicar cascata
SENÃO Aplicar impedimento OU Aplicar cascata OU Aplicar anulação (caso 1)

2.4 Modelo Orientado a Objetos

A indústria da informática, desde seu início, recria no computador cópias bidimensionais do mundo em que vivemos. Começamos com arquivos, registros, relacionamentos e chegamos aos objetos de hoje [RUMB91].

O objetivo continua o mesmo até hoje: representar da maneira mais fiel possível o mundo que nos cerca. É através desta idéia que a Orientação a Objetos vem superando em alguns pontos o modelo Relacional, pois consegue representar de forma mais natural os objetos do mundo real, facilitando assim toda a cadeia de desenvolvimento de software.

Superficialmente, pode-se dizer que orientação a objetos corresponde à organização de sistemas como uma coleção de objetos que integram estruturas de dados e comportamento. Além desta noção básica, a abordagem inclui um certo número de conceitos, princípios e mecanismos que a diferenciam das demais. Seus principais conceitos são apresentados em seguida[FERN8I].

Abstração: é a consideração apenas das propriedades comuns de um conjunto de objetos, omitindo os detalhes, utilizada com freqüência na definição de valores similares e na formação de um tipo a partir de outro, em diferentes níveis de abstração. O uso de abstrações permite a geração de tipos baseada em hierarquias de tipos e de relacionamentos.

Os principais conceitos de abstração utilizados em banco de dados são generalização e agregação. A generalização corresponde à associação "é um" onde, a partir de propriedades comuns de diferentes entidades, é criada uma outra entidade. O processo inverso é a especialização. A agregação corresponde à associação "parte de".

Classe: Um conjunto de objetos que possui o mesmo tipo (atributos, relacionamentos, operações) pode ser agrupado para formar uma classe. A noção de classe é associada ao tempo de execução, podendo ser vista como uma representação por extensão, enquanto que o tipo é uma representação intencional. Cada classe tem um tipo associado, o qual especifica a estrutura e o comportamento de seus objetos. Assim, a extensão da classe denota o conjunto dos objetos atualmente existentes na classe e o tipo provê a estrutura destes objetos.

Classe abstrata: Uma classe que não tem instâncias e sim subclasses que, por sua vez, podem ter instâncias.

Objeto: Os objetos são abstrações de dados do mundo real, com uma interface de nomes de operações e um estado local que permanece oculto. As abstrações da representação e das operações são ambas suportadas no modelo de dados orientado a objetos, ou seja, são incorporadas as noções de estruturas de dados e de comportamento. Um objeto tem um estado interno descrito por atributos que podem apenas ser acessados ou modificados através de operações definidas pelo criador do objeto. Um objeto individual é chamado de instância ou ocorrência de objeto. A parte estrutural de um objeto (em banco de dados) é similar à noção de entidade no modelo Entidade-Relacionamento.

Identidade de Objeto: Num modelo com identidade de objetos, estes têm existência independente de seus valores correntes e dos endereços de armazenamento físico. A identidade do objeto é geralmente gerada pelo sistema. A impossibilidade de garantir a identificação de objetos exclusivamente através de suas propriedades estruturais e comportamentais motivou a definição de identificadores únicos de objetos, que persistem no tempo de forma independente ao estado interno do objeto.

A identidade de objetos elimina as anomalias de atualização e de integridade referencial, uma vez que a atualização de um objeto será automaticamente refletida nos objetos que o referenciam e que o identificador de um objeto não tem seu valor alterado.

Atributos: Tipos de dados predefinidos ou definidos pelo usuário. Os atributos modelam a estrutura da entidade do mundo real.

Métodos: Uma rotina (procedure ou function) criada como um atributo de um objeto. Possui visibilidade dos atributos do objeto e pode operá-los diretamente. São diferentes de *Stored Procedures* em dois aspectos: execução e acesso aos dados. Sua execução está ligada a um objeto (devemos fazer referência a ele); além disso, ele possui completo acesso aos atributos de seu objeto.

Encapsulamento: Mecanismo através do qual dados e operações são vistos como um único objeto.

Herança: É um mecanismo que permite ao usuário definir tipos de forma incremental, por refinamento de outros já existentes, permitindo composição de tipos em que as propriedades de um ou mais tipos são reutilizadas na definição de um novo tipo. De fato, ela corresponde à transferência de propriedades estruturais e de comportamento de uma classe para suas subclasses.

As principais vantagens de herança são prover uma maior expressividade na modelagem dos dados, facilitar a reusabilidade de objetos e definir classes por refinamento, podendo fatorar especificações e implementações como na adaptação de métodos gerais para casos particulares, redefinindo-os para estes, e simplificando a evolução e a reusabilidade de esquemas de banco de dados.

Herança Simples: Na herança simples um certo tipo pode ter apenas um supertipo, da mesma forma uma subclasse só herda diretamente de uma única classe. Podemos classificar esta herança em quatro subtipos: de substituição, de inclusão, de restrição e de especialização.

Herança Múltipla: Nesta herança um tipo pode ter supertipos e os mesmos refinamentos de herança simples. Há basicamente dois tipos de conflitos referentes à herança múltipla: entre o tipo e o supertipo e entre múltiplos supertipos. O primeiro pode ser resolvido dando-se prioridade à definição presente no tipo, e não no supertipo. Com os conflitos entre múltiplos supertipos, como uma resolução por default pode causar heranças não desejadas, a abordagem mais segura é baseada na requisição explícita da intervenção do usuário.

Polimorfismo: Em sistemas polimórficos uma mesma operação pode se comportar de diferentes formas em classes distintas. Como exemplo temos a operação print que será implementada de forma diferente se o objeto correspondente for um texto ou uma imagem: dependendo do objeto teremos um tipo de impressão. Tem-se também polimorfismo quando ocorre a passagem de diferentes tipos de objetos como parâmetros enviados a outros objetos

Um mesmo nome pode ser usado por mais de uma operação definida sobre diferentes objetos, o que caracteriza uma sobrecarga. A redefinição do operador para cada um dos tipos de

objetos definidos caracteriza uma sobreposição. As operações são ligadas aos programas em tempo de execução caracterizando o acoplamento tardio ou *late binding*.

2.5 Resumo

Este capítulo apresentou alguns conceitos básicos sobre as funcionalidades de um Sistema de Faturamento aplicado em *Telecom*, para nos familiarizarmos com as informações tratadas pelo módulo EBV. Demonstrou também todas as capacidades do protótipo, o qual será submetido à aplicação prática deste trabalho. Vimos ainda alguns aspectos básicos dos modelos: Relacional e Orientado a Objetos, com o intuito de colocar ao leitor a estrutura básica dos modelos que deram origem ao modelo Objeto-Relacional, que será apresentado no próximo capítulo.

Capítulo 3

Modelo Objeto-Relacional

Os sistemas de gerenciamento de banco de dados evoluíram de modelos Hierárquicos para modelos de Rede e, posteriormente, para modelos Relacionais[FERN8I].

Mas com as novas necessidades impostas pelo desenvolvimento das tecnologias Orientadas a Objetos, o Modelo Relacional tornou-se desatualizado, com isto alguns fabricantes estenderam novas funcionalidades criando o Modelo Objeto-Relacional . O novo modelo é baseado na idéia de estender o modelo Relacional, fornecendo um sistema de tipos mais rico, através da inclusão de características da Orientação a Objetos e adicionando construções às linguagens de consultas Relacionais, tal como SQL para manipular os novos tipos de dados adicionados [SILV98]. Tais extensões tentam preservar os fundamentos Relacionais, enquanto estendem o poder de modelagem dos dados. Entretanto, ainda não se pode afirmar que existe um modelo Objeto Relacional aceito como padrão. Os SGBDs baseados neste modelo sofrem o mesmo problema que os SGBDs Orientados a Objetos, isto é, há muitas diferenças nos produtos disponíveis no mercado.

Sendo assim, a partir de fundamentações teóricas [SM96, SK98, EM99] e em modelos implementados por produtos comerciais, apresentamos nas próximas seções, os principais conceitos teóricos do modelo Objeto-Relacional e alguns exemplos de implementação, utilizando as sintaxes propostas pela Oracle na versão 8i.

O SGBD Oracle 8i foi escolhido como objeto de estudo porque é utilizado pelo protótipo de visualização de contas EBV, apresentado no capítulo anterior, que na realidade será nosso alvo para a aplicação prática dos estudos deste capítulo.

3.1 Object Type

O conceito de *Object Type* é equivalente ao de classes em Orientação a Objetos, ou seja, declaração de atributos, declaração de métodos e corpo dos métodos.

Object Type é composto por[ORA9I]:

- Um *nome* que identifica o objeto unicamente dentro do esquema;
- Um ou mais atributos que modelam a estrutura do objeto (podem ser tipos primitivos ou definidos pelo usuário como um Varray, Nested Table ou outro objeto) ;
- Zero ou mais métodos que são funções ou procedimentos implementados em alguma linguagem suportada pelo SGBD e armazenados no banco.

Um *Object Type* é uma abstração de entidades do mundo real que:

- Não permite nenhum tipo de armazenamento (ele não é uma tabela) ;
- Deve ser visto como mais um tipo de dado disponível que pode ser utilizado na definição de uma TABELA (armazenamento de instâncias de objetos em tabelas) ou uma COLUNA de uma tabela.

Uma tabela criada a partir de um *Object Type* é denominada tabela de objetos (*OBJECT TABLE*), cada linha armazenada nesta tabela é denominada linha objeto (*OBJECT ROW*) e cada linha tem um OID (*Object Identifier*) associado.

Uma coluna cujo tipo de dado é um Object Type é denominada uma coluna objeto (*OBJECT COLUMM*) e não possui um OID.

No exemplo a seguir, criaremos um Object Type *OBJ_END* que será usado como atributo de um outro Object Type[FERN8I].

```
CREATE TYPE  OBJ_END AS OBJECT
(NM_RUA             VARCHAR2(60),
 NM_BAIRRO          VARCHAR2(60),
 NM_CIDADE          VARCHAR2(60),
 CD_ESTADO          VARCHAR2(2),
 NR_RUA             NUMBER(8),
 NR_CEP             NUMBER(8));

Type created
```

```
CREATE TYPE OBJ_CLI AS OBJECT
(NR_CARTAO          NUMBER(3),
 NM_CLIENTE         VARCHAR2(60),
 NM_ENDERECO OBJ_END);

Type created
```

Como podemos ver, o atributo *NM_ENDERECO* do Object Type *OBJ_CLI* é do tipo *OBJ_END* criado anteriormente, portanto este atributo passa a ter a mesma estrutura do Object Type que o derivou.

3.2 Object Table

Existem várias formas de utilizar o Object Type, nesta seção descreveremos como criar tabelas utilizando este conceito.

Uma das formas de utilizar o Object Type para a criação de uma tabela seria criá-la no escopo Relacional contendo colunas cujo tipo é um objeto. A outra é criarmos uma Object Table, ou seja, uma tabela capaz de armazenar as instâncias de uma determinada classe [KYTE8I].

Uma instância de uma classe é o resultado da construção de um elemento com características definidas pela classe, ou seja, da construção de um objeto.

Para os exemplos apresentados a seguir, utilizaremos os Object Types criados na seção anterior.

Criação da tabela Relacional:

```
CREATE TABLE TCLI
(NR_CARTAO          NUMBER(3),
 NM_CLIENTE         VARCHAR2(60),
 NM_ENDERECO        OBJ_END);

Table created.
```

No exemplo acima, criamos uma tabela Relacional contendo uma coluna definida por um Object Type, com isto teremos uma tabela Relacional contendo uma coluna com a estrutura de um objeto.

Criação da tabela Objeto:

```
CREATE TABLE CLIENTE OF OBJ_CLI
(PRIMARY KEY                          (NR_CARTAO),
 NM_ENDERECO                           NOT NULL,
 CHECK (NM_ENDERECO.NM_RUA             IS NOT NULL),
 CHECK (NM_ENDERECO.NM_BAIRRO          IS NOT NULL),
 CHECK (NM_ENDERECO.NM_CIDADE          IS NOT NULL),
 CHECK (NM_ENDERECO.NR_CEP             IS NOT NULL,
 CHECK (NM_ENDERECO.CD_ESTADO          IS NOT NULL));

Table created.
```

No exemplo acima criamos uma tabela para armazenar objetos do tipo *OBJ_CLI* criado anteriormente. Observe que na criação dos tipos não são estabelecidos critérios de restrição ou crítica.

As restrições são estabelecidas somente na criação da tabela. Isto porque indicamos que as instâncias do Objeto *OBJ_CLI*, quando armazenadas na tabela *CLIENTE*, serão restringidas pelas regras estabelecidas.

Não podemos nos esquecer, porém, que os objetos definidos na seção 3.1 podem ser reutilizados em outras definições de tabelas, em variáveis de PL/SQL, como atributos de outros objetos, e assim por diante, com restrições diferenciadas[ORA9I].

Inserindo dados na tabela Objeto:

```
INSERT INTO CLIENTE ( NR_CARTAO, NM_CLIENTE, NM_ENDERECO)
VALUES ( 100, 'ROGÉRIO P. COLFERAI', OBJ_END ( 'AV JOAO MARQUES', 'JD
SANTA MARTA', 'CAMPINAS', 'SP', 1234, 13970000)
/
1 row created
```

Observe que a referência ao endereço também é feita com o nome do tipo correspondente, porém há necessidade de informarmos todos os atributos. Quando não desejamos completar alguma informação, devemos, explicitamente, preencher com NULL[ORA91].

Alteração nos dados da tabela Objeto:

```
UPDATE CLIENTE C
SET C.NM_CLIENTE                = 'MARCOS JOSE DA SILVA',
    C.NM_ENDERECO.NM_RUA        = 'RUA MATHEUS FILHO',
    C.NM_ENDERECO.NM_BAIRRO     = 'CENTRO',
    C.NM_ENDERECO.NR_RUA        = 245,
    C.NM_ENDERECO.NR_CEP        = 13978002
WHERE NR_CARTAO = 100;

1 row updated
```

No exemplo acima, efetuamos alterações em colunas do objeto *OBJ_CLI* e do objeto *OBJ_END*.

Como regra, temos que a referência a um atributo de um objeto deve ser feita com o nome do objeto correspondente.

Uma vez que o objeto *OBJ_CLI* é o próprio objeto armazenado na tabela *CLIENTE*, a referência aos seus atributos é feita apenas com a utilização do alias da tabela, que deve ser mencionado obrigatoriamente. A utilização do alias da tabela se repetirá todas as vezes que desejarmos fazer um referência individual a um atributo específico do tipo objeto armazenado na tabela[ORA9I]. Para referenciarmos os atributos do objeto interno, devemos utilizar o nome da coluna, que neste caso foi *NM_ENDERECO* do tipo *OBJ_END*, a qual contém diversos atributos.

Consulta em uma tabela Objeto:

```
SELECT  NR_CARTAO,
        NM_CLIENTE,
        C.NM_ENDERECO.NM_RUA,
        C.NM_ENDERECO. NR_RUA
FROM    CLIENTE C
WHERE NR_CARTAO IN (100,200);
```

Neste exemplo, efetuamos a obtenção de apenas algumas colunas do tabela objeto CLIENTE. Observe que a referência aos atributos simples não necessitou da menção do alias da tabela, porém, quando desejarmos obter algum atributo do objeto interno, como neste caso, algum atributo do objeto OBJ_END, houve a necessidade de utilizarmos tanto o alias da tabela quanto o nome da coluna.

```
SELECT  VALUE(C)  FROM CLIENTE C;
```

Neste segundo exemplo, todos os dados do objeto OBJ_CLI serão apresentados, sem que tenhamos a necessidade de definir qualquer atributo. Podemos considerar VALUE como uma função que retorna a linha objeto.

3.3 Relacionamentos

Em um modelo Relacional, os relacionamentos estabelecem restrições que participam das regras de negócio estabelecidas no banco de dados.

No modelo Objeto Relacional, utilizamos referência (*REF*), para estabelecer uma ligação com um objeto específico, ou mais formalmente, com uma instância específica de um objeto.

A expressão *REF* difere de um relacionamento estabelecido num modelo Relacional, pois ela não estabelece uma restrição de integridade, não há impedimento à remoção de uma instância que esteja sendo referenciada por outra. A expressão *REF* corresponde a um ponteiro, objetiva acesso mais rápido e não regra de integridade[FERN8I].

Nos exemplos a seguir mostraremos o uso da expressão REF.

```
CREATE TYPE OBJ_CLIENTE AS OBJECT
(NR_CARTAO          NUMBER(3),
 NM_CLIENTE         VARCHAR2(60),
 NR_TEL             VARCHAR2(20));

Type created

CREATE TABLE CLIENTE OF OBJ_CLIENTE
(PRIMARY KEY        (NR_CARTAO),
 NM_CLIENTE          NOT NULL);

Table created

CREATE OR REPLACE TYPE OBJ_DEPENDENTE AS OBJECT
(NM_DEPENDENTE           VARCHAR2(60),
 NR_CARTAO               NUMBER(3),
 RF_TITULAR              REF OBJ_CLIENTE);

Type created
```

```
CREATE TABLE DEPENDENTE OF OBJ_DEPENDENTE
(PRIMARY KEY              (NR_CARTAO),
 RF_TITULAR               SCOPE IS CLIENTE,
 NM_DEPENDENTE            NOT NULL);

Table created
```

Inicialmente criamos o tipo de objeto OBJ_CLIENTE, uma tabela para armazenamento dos objetos deste tipo, com as restrições necessárias.

Em seguida criamos o tipo objeto OBJ_DEPENDENTE contendo um atributo RF_TITULAR cujo tipo é REF; isto significa que este atributo estará fazendo uma referência e não armazenando dados escalares[FERN8I]. Esta referência poderá ser feita para objetos do tipo OBJ_CLIENTE que estejam armazenados em qualquer tabela.

Quando implementamos a tabela DEPENDENTE para armazenamento de objetos do tipo OBJ_DEPENDENTE, limitamos a referência da coluna RF_TITULAR a objetos armazenados na tabela CLIENTE.

A seguir mostraremos um exemplo de como inserir e relacionar os objetos nas tabelas criadas no exemplo anterior.

```
INSERT INTO CLIENTE VALUES( 1,'CLIENTE1','1937061212');
1 row created

INSERT INTO CLIENTE VALUES( 2,'CLIENTE2','1937063434');
1 row created

INSERT INTO DEPENDENTE
            SELECT  'DEPENDENTE1', 431,  REF(C)
            FROM CLIENTE C
            WHERE NR_CARTAO = 1;
1 row created

INSERT INTO DEPENDENTE
            SELECT  'DEPENDENTE2', 432,  REF(C)
            FROM CLIENTE C
            WHERE NR_CARTAO = 2;
1 row created
```

Para armazenar as referências ao objeto armazenado em CLIENTE, precisamos realizar uma consulta para obter o OID (Object Identifier) do objeto desejado. O OID é uma informação acrescida pela Oracle a todos os objetos, para garantir a unicidade dos mesmos. Observe, ainda, que para observarmos a indicação do objeto usamos, novamente, a função *REF* do objeto apelidado de C.

No próximo exemplo mostraremos como consultar via referência.

```
SELECT  D.RF_TITULAR.NR_CARTAO,
        D.RF_TITULAR.NM_CLIENTE,
        NM_DEPENDENTE, NR_CARTAO
FROM DEPENDENTE D ;
```

Utilizamos no exemplo acima a coluna RF_TITULAR (ou o atributo do objeto DEPENDENTE armazenado na tabela DEPENDENTE) para obter informações diretamente dos objetos a que ela faz referência. Observe que apelidamos nosso objeto de D e solicitamos que fosse obtido o número do cartão através da referência RF_TITULAR do objeto D que foi selecionado.

A cláusula *REF*, por si só, não garante a integridade referencial, mas isto não quer dizer que não possamos ter esta garantia. Podemos optar entre referência sem integridade ou com integridade, de acordo com a situação. A referência pode ser usada apenas como meio de acesso direto[ORA9I].

Mostraremos abaixo um exemplo de como é possível garantir a integridade referencial entre os objetos. Utilizaremos os mesmos objetos dos exemplos mostrados acima, mas recriaremos a tabela objeto DEPENDENTE.

```
CREATE TABLE DEPENDENTE OF OBJ_DEPENDENTE
(PRIMARY KEY              (NR_CARTAO),
 RF_TITULAR                WITH ROWID REFERENCES CLIENTE,
 NM_DEPENDENTE             NOT NULL);

Table created
```

Vamos observar as diferenças na criação da tabela DEPENDENTE. Quando estabelecemos as restrições para a coluna RF_TITULAR, determinamos duas cláusulas: ***WITH ROWID*** e ***REFERENCES***.

A cláusula WITH ROWID tem a finalidade de importar, do objeto associado, não apenas o OID, mas também a identificação física da linha. Isto dará, certamente, muito mais velocidade de acesso quando desejarmos obter informações do objeto referenciado, uma vez que contamos com o endereço da linha para acesso direto[ORA9I, FERN8I].

A cláusula REFERENCES garante a integridade referencial, ou seja, não poderemos remover elementos de CLIENTE se houver dependentes relacionados. Poderíamos ter

determinado que na remoção de clientes, os dependentes fossem removidos simultaneamente ou ainda, que na remoção de cliente o relacionamento dos dependentes fosse anulado.

3.4 Coleções

Coleções são mecanismos que permitem o agrupamento de elementos de mesmo tipo e podem ser constituídas de tipos primitivos, tipos objetos ou referências. Existem dois tipos de coleções: *Varray* e *Nested Tables* [ORA9I].

3.4.1 Varray Type

Varray é um conjunto de elementos ordenado e limitado que pode ser utilizado como um tipo de dado de uma coluna, como um atributo de um *Object Type* ou como variável, parâmetro ou tipo resultante de uma função em um bloco de código. Caso ele seja um tipo de dado de uma coluna de uma tabela, seus elementos são armazenados na própria tabela [SALG99].

Exemplo:

```
CREATE TYPE   ARR_TEL  AS VARRAY(5) OF VARCHAR2(9);

Type created.


CREATE TABLE CLIENTE
(NR_CARTAO          NUMBER(03),
 NM_CLIENTE         VARCHAR2(60),
 NR_TEL             ARR_TEL,
 PRIMARY KEY (NR_CARTAO));

Table created.
```

3.4.2 Nested Table

Nested Table é um conjunto de elementos não ordenado e ilimitado que pode ser utilizado como um tipo de dado de uma coluna da tabela, como atributo de um Object Type ou como uma variável, parâmetro ou tipo resultante de uma função em um bloco de código. Caso ele seja um tipo de dado de uma coluna de uma tabela, seus elementos são armazenados em uma outra tabela associada à tabela hospedeira [SALG99].

O tipo de dado de uma Nested Table pode ser um escalar (varchar2, number, date, char, etc.), uma referência a um objeto ou um tipo objeto.

Exemplo:

```
CREATE TYPE  NT_TEL AS TABLE OF VARCHAR2(9);

Type created.

CREATE TABLE CLIENTE
(NR_CARTAO          NUMBER(03),
 NM_CLIENTE         VARCHAR2(60),
 NR_TEL             NT_TEL,
 PRIMARY KEY (NR_CARTAO))
 NESTED TABLE NR_TEL STORE AS TAB_NR_TEL;
Table created.
```

No exemplo estabelecemos a Nested Table como uma coluna da tabela relacional CLIENTE e definimos que seu armazenamento será realizado na tabela física TAB_NR_TEL.

Quando usar Varrays ?

Varrays e Nested Tables são alternativas quando temos informações repetitivas.

Um Varray é uma escolha melhor quando:

- A ordem de indexação é importante. Um Varray é uma coleção ordenada (por um indexador), enquanto uma Nested Table não é.
- O número de elementos é conhecido e pode ser limitado. O Varray força a especificação de um número limite de elementos. Seu armazenamento pode ser mais

eficiente uma vez que a quantidade de ocorrências é conhecida, enquanto Nested Table não são limitadas.

- Não há necessidade de efetuarmos consultas freqüentes a um elemento do Varray. As ocorrências do Varray são tratadas como um único elemento, sendo assim o acesso individual ao elemento deve ser feito através de rotinas PL/SQL. Havendo necessidade de restrições ou atualizações individuais, a Nested Table pode ser preferível. Caso pretendamos recuperar a coleção como um todo, o Varray é mais eficiente.

Quando usar Nested Tables ?

Uma Nested Table é uma escolha melhor do que um Varray quando:

- A consulta a linhas específicas dos dados é indispensável. Este processo pode ser executado para Varray, porém perdemos em eficiência pois temos que usar o recurso (CAST) para a transformação de Varray para Nested Table primeiro.
- A Indexação de colunas de uma Nested Table pode ser necessária. Este recurso não é possível em Varray.
- A ordem de cadastramento das linhas não é importante, e caso haja alguma necessidade de ordem podemos provê-la através de um índice em uma coluna da tabela embutida.
- Não temos idéia da quantidade de linhas da tabela embutida, ou não há limite a considerar. A definição de um Varray necessita de um limite definido que seja real para que possamos lucrar no armazenamento.

3.5 Métodos

Um método, como sabemos, é um código associado (encapsulado) ao objeto. Sua principal característica é ter acesso ilimitado aos dados do objeto ao qual está ligado.

Quando criamos um tipo de objeto, podemos associar a este tipo rotinas, as quais têm acesso a todos os atributos do tipo diretamente. Estas rotinas são referenciadas como se fossem atributos do tipo e são chamadas de Métodos[FERN8I].

A criação das rotinas obedece a duas etapas; a primeira se refere à parte de especificação da rotina, que ocorre juntamente com a criação do tipo de objeto, onde definiremos o nome da rotina, os parâmetros e o tipo de retorno no caso de função.

Após a definição da especificação, passamos à definição do corpo da rotina. Para tal, criaremos um Type Body associado ao tipo definido. Desta forma, todas as rotinas associadas àquele tipo serão definidas no corpo do tipo[ORA9I].

Existem três tipos de métodos que podemos associar a um tipo de objeto[FERN8I].

MAP MEMBER – é uma função que retorna uma informação escalar (number, varchar2, char, etc.) com a finalidade de determinar a posição relativa do objeto particular no conjunto.

Esta função pode ser acionada para estabelecer comparação entre os objetos, como ordenação por exemplo.

ORDER MEMBER – especifica uma função que recebe como parâmetro um outro objeto deste mesmo tipo.

A PL/SQL usa a ordenação para avaliar expressões boleanas do tipo >(maior) e <(menor) e para realizar comparações implícitas tais como distinct, Group By e cláusulas Order By. O método MAP retorna a posição relativa de um objeto em relação ao grupo.

Outra forma de estabelecer ordem é a utilização de um método do tipo ORDER que, diferentemente do método MAP, deve receber como parâmetro outro objeto do mesmo tipo no qual o método está sendo criado a fim de que internamente na rotina estabeleça as condições de comparação e retorne a indicação de qual dos dois objetos (aquele recebido como parâmetro ou o próprio objeto) é maior, menor ou igual.

MEMBER – especifica um procedimento ou função associada com o objeto e que é referenciada como um atributo. Esta rotina deve ser acionada explicitamente pelas aplicações ou nos comandos de SQL.

3.6 Visões de Objetos

Uma visão de objetos é uma tabela de objetos virtual, que são baseadas em consultas a tabelas relacionais ou de objetos. Elas permitem que novas aplicações orientadas a objetos sejam construídas utilizando esquemas e dados relacionais[ORA9I].

A criação das views a fim de simular tabelas objetos permite um ambiente de transição entre as tabelas relacionais e as objetos.

As aplicações já escritas continuam vendo o ambiente relacional, enquanto as novas aplicações podem considerar a existência de objetos utilizando, no entanto, o mesmo conjunto de tabelas[FERN8I].

As operações permitidas são inserções, atualizações e deleções (quando não possuem junções ou funções de grupo).

Visões com junções ou funções de grupo podem ser atualizadas através de gatilhos (TRIGGERS) do tipo INSTEAD OF, que sobrepõem as operações de inserção, atualização e deleção[ORA9I].

Para demonstrar algumas funcionalidades, criaremos tabelas relacionais, em seguida, criaremos views com base nestas tabelas.

Criação das tabelas relacionais:

```
CREATE TABLE RCLIENTE (
 NR_CARTAO                NUMBER(3),
 NM_CLIENTE               VARCHAR2(60,
 NR_DDD1                  CHAR(3),
 NR_TEL1                  NUMBER(9),
 NR_DDD2                  CHAR(3),
 NR_TEL2                  NUMBER(9),
 NR_DDD3                  CHAR(3),
 NR_TEL3                  NUMBER(9),
 PRIMARY KEY (NR_CARTAO));

TABLE CREATED.
```

Criamos uma tabela relacional chamada RCLIENTE contendo: nome, número do cartão e três ocorrências de telefone. Os endereços serão armazenados em tabela separada e relacionados com a tabela RCLIENTE.

```
CREATE TABLE RENDERECO (
 NM_RUA                    VARCHAR2(60),
 CD _ESTADO                CHAR(2)  REFERENCES ESTADOS,
 NR_RUA                    NUMBER(8),
 NR_CARTAO                 NUMBER(3),
 SQ_ENDERECO               NUMBER(2),
 PRIMARY KEY (NR_CARTAO, SQ_ENDERECO),
 FOREIGN KEY (NR_CARTAO) REFERENCES RCLIENTE);

TABLE CREATED.

CREATE TABLE RDEP(
 NR_CARTAO_TIT             NUMBER(3),
 NR_CARTAO                 NUMBER(3),
 NM_DEPENDENTE             VARCHAR2(60),
 PRIMARY KEY (NR_CARTAO),
 FOREIGN KEY ((NR_CARTAO_TIT) REFERENCES RCLIENTE);

TABLE CREATED.
```

Criamos duas tabelas relacionais, RENDERECO e RDEP, para guardar os dados de endereço associados ao cliente e aos dependentes do cliente.

Para prepararmos a visualização das tabelas relacionais como objetos, iniciaremos criando os tipos a serem utilizados na View.

Criação dos tipos objetos utilizados pela View:

```
CREATE TYPE OTEL AS OBJECT (
 NR_DDD                    CHAR(3),
 NR_TEL                    NUMBER(9));

TYPE CREATED.


CREATE TYPE VR_OTEL AS VARRAY(3) OF OTEL;
TYPE CREATED.
CREATE TYPE OEND AS OBJECT(
 NM_RUA                    VARCHAR2(60),
 CD_ESTADO                 VARCHAR2(2),
 NR_RUA                    NUMBER(8));

TYPE CREATED.

CREATE TYPE NT_OEND AS TABLE OF OEND;

TYPE CREATED.
```

Criamos acima, um objeto OTEL contendo as informações básicas de telefone. Este objeto foi utilizado na criação do tipo Varray VR_OTEL.

De forma similar, criamos um objeto OEND contendo as informações básicas de endereço. Este objeto foi utilizado na criação do tipo Nested Table NT_OEND.

```
CREATE TYPE OCLIENTE AS OBJECT(
 NR_CARTAO                NUMBER(3),
 NM_CLIENTE               VARCHAR2(60),
 NR_TEL                   VR_OTEL,
 NM_ENDERECO              NT_OEND);

TYPE CREATED.

CREATE TYPE ODEP AS       OBJECT(
 NR_CARTAO                NUMBER(3),
 NM_DEPENDENTE            VARCHAR2(60),
 RF_TITULAR               REF OCLIENTE);

TYPE CREATED.
```

Acima criamos o tipo objeto OCLIENTE já contendo os atributos NR_TEL do tipo VR_OTEL e NM_ENDERECO do tipo NT_OEND.

Criamos também o objeto ODEP referente ao dependente contendo uma referência REF ao objeto cliente.

Criação da View Objeto:

```
CREATE OR REPLACE VIEW VCLIENTE OF OCLIENTE
       WITH OBJECT OID(NR_CARTAO) AS
 SELECT RC.NR_CARTAO, RC.NM_CLIENTE,
          VR_OTEL(OTEL(RC.NR_DDD1, RC.NR_TEL1),
                       OTEL(RC.NR_DDD2, RC.NR_TEL2),
          OTEL(RC.NR_DDD3, RC.NR_TEL3))
       CAST(MULTISET( SELECT RE.NM_RUA, RE.CD_ESTADO,
                             RE.NR_RUA
                             FROM RENDERECO RE
                             WHERE  RE.NR_CARTAO = RC.NR_CARTAO)
       AS NT_OEND)
             FROM RCLIENTE RC;

VIEW CREATED.
```

A View criada acima, seleciona basicamente dados da tabela relacional RCLIENTE e os formata na mesma ordem do objeto OCLIENTE. Iniciamos com os atributos escalares NR_CARTAO E NM_CLIENTE. Para o Varray NR_TEL, fizemos a montagem utilizando os métodos construtores implícito VR_OTEL do Varray e OTEL do objeto. Para a Nested Table

NM_ENDERECO, fizemos a leitura da tabela RENDERECO, trazendo os dados na ordem do objeto OEND. A expressão MULTISET indicou que diversas linhas seriam retornadas da subquery. A expressão CAST acomodou os dados ao layout do objeto OEND na Nested Table NT_OEND. Observe que apenas as linhas referentes ao cartão selecionado na query externa foram trazidas; para tal, utilizamos subselect correlacionado[BP98].

```
CREATE OR REPLACE VIEW VDEP OF ODEP
       WITH OBJECT OID (NR_CARTAO) AS
  SELECT RD.NM_CARTAO, NM_DEPENDENTE,
           MAKE_REF(VCLIENTE, RD.NR_CARTAO_TIT)
  FROM RDEP RD;

VIEW CREATED.
```

Na View apresentada acima, utilizamos a função MAKE_REF para estabelecer a referência dinâmica da View VCLIENTE relativa ao cartão NR_CARTAO_TIT.

Em ambos os exemplos, a criação da View precisou da expressão WITH OBJECT OID a fim de determinar qual o atributo ou conjunto de atributos que devem ser usados para identificar univocamente o registro da View [GZS00].

3.7 Resumo

Este capítulo apresentou as principais características de implementação do modelo Objeto Relacional proposto pela Oracle na versão 8i, buscando deixar o leitor familiarizado com os recursos, a fim de facilitar o entendimento do restante do trabalho.

O próximo capítulo apresenta dois modelos relacionais, o primeiro é uma parte de um modelo mais complexo, que suporta um sistema de Faturamento do qual originou recursos que foram utilizados para a criação do segundo modelo apresentado, que hoje suporta uma aplicação Orientada a Objetos, que é o nosso módulo de visualização de contas EBV demonstrado no capítulo anterior.

Capítulo 4

Esquema Relacional EBV

O objetivo deste capítulo é demonstrar o esquema de dados relacional que suporta atualmente o protótipo do módulo EBV. Este esquema nos servirá de base para aplicação prática de alguns conceitos do modelo Objeto-Relacional, que foram demonstrados no capítulo anterior.

O capítulo está organizado da seguinte forma. A seção 4.1 apresenta parte de um esquema de dados que deu origem à estrutura que atualmente suporta o protótipo do visualizador de contas EBV. Na seção 4.2 analisaremos em detalhes o esquema de dados criado para suportar o protótipo. A seção 4.3 apresenta um resumo do capítulo.

4.1 Esquema Original do Sistema de Faturamento

O esquema que denominamos origem foi extraído de uma complexa estrutura de dados relacional, ao qual, suporta um Sistema de Faturamento. Este escopo de origem embasou as regras de negócio na elaboração de um esquema de dados, que atualmente suporta o protótipo do módulo EBV. O esquema contempla o cadastro completo dos clientes e itens presentes nas contas, como ligações e serviços prestados pela operadora, agrupados e classificados devidamente pelo tipo de grupo de serviço a que pertencem. Estes itens são faturados dentro de um determinado período de tempo, ao qual chamamos de período de faturamento.

A estrutura armazena ainda documentos, ou seja, as contas de cada meio de acesso como telefone fixo, móvel, TV a cabo e outros meios oferecidos pela empresa operadora, apropriados a seus clientes responsáveis.

Além destas informações, que de modo geral são os dados utilizados pelo módulo EBV, o esquema nos traz uma grande quantidade de informações extras, que são utilizadas pelo Sistema de Faturamento. Estas informações seriam prejudiciais ao protótipo, pois causaria uma sobrecarga de dados e poderia comprometer o desempenho das consultas à base de dados, visto que o desempenho é um fator muito importante para este tipo de serviço.

Não podemos esquecer que além de ter influenciado na estrutura e nas regras de negócio da construção do esquema de dados do módulo EBV, o escopo de origem também é fonte geradora de dados, sendo que ele foi extraído do Sistema de Faturamento ao qual o protótipo se aplica.

A figura 4.1 apresenta o escopo da estrutura que deu origem ao esquema de dados do módulo EBV.

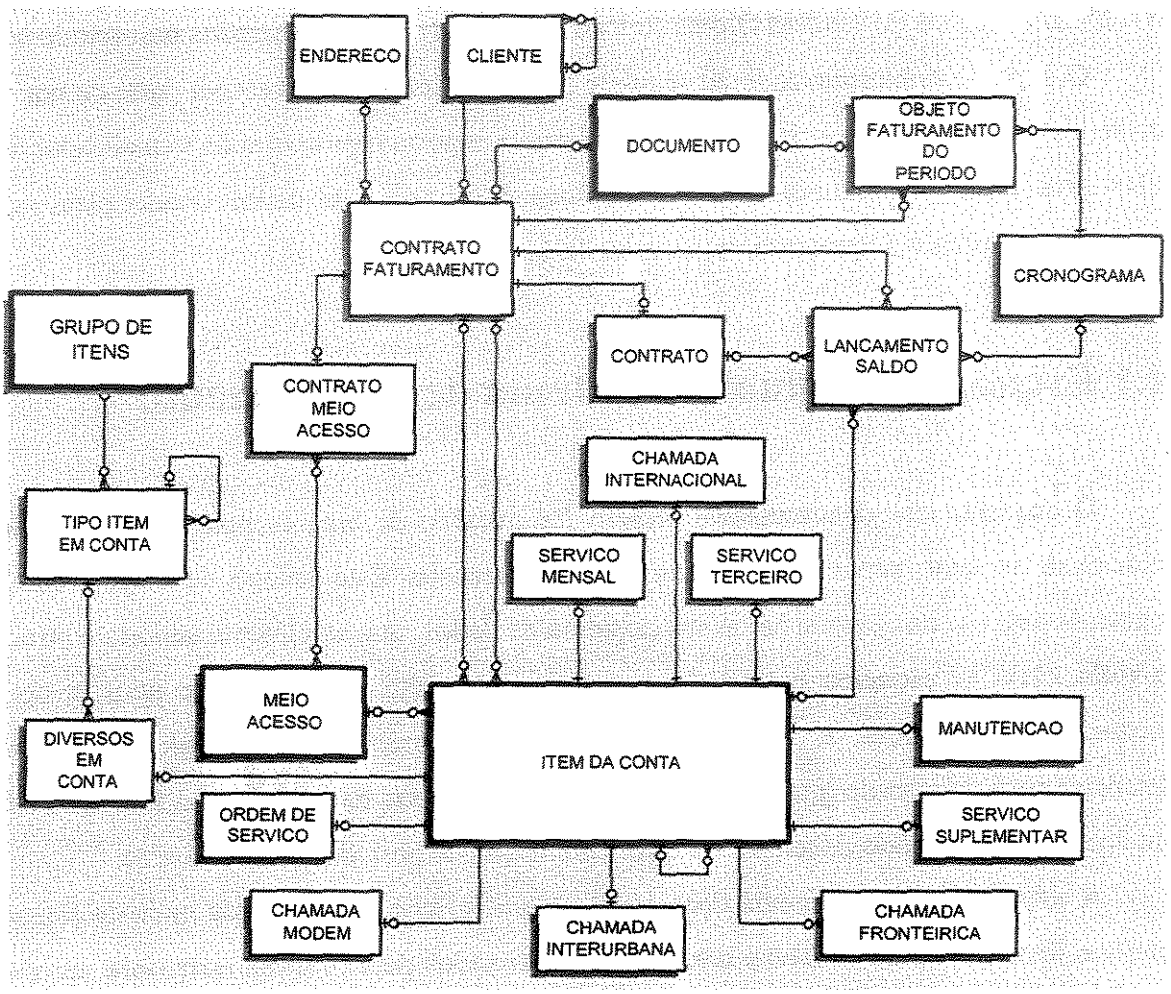

Figura 4.1: Esquema Relacional Origem

4.1.1 Descrição das Entidades

Apresentamos abaixo uma breve descrição de cada entidade contida no esquema apresentado na figura 4.1.

GRUPO DE ITENS

Grupo de modalidade de serviço por empresa prestadora, especificando blocos de serviço, para tratamento diferenciado. Relaciona tipos de itens de conta pertencentes a uma modalidade de serviço e/ou a uma empresa prestadora. A conta (documento de cobrança ao cliente) será emitida considerando-se os blocos de serviço definidos sobre sua ordem de prioridade para exibição.

CHAMADA MODEM

Entidade de Comunicação de Dados. Subtipo de Item da Conta para o Serviço de Cable Modem.

CHAMADA INTERURBANA

Subtipo de Item da Conta que identifica os atributos referentes às ligações interurbanas.

CHAMADA INTERNACIONAL

Subtipo de Item da Conta que identifica os atributos referentes às ligações de tráfego internacional.

CHAMADA FRONTEIRIÇA

Subtipo de Item da Conta que identifica os atributos referentes às ligações de tráfego fronteiriço.

CLIENTE

Pessoa física ou jurídica que utiliza/utilizou um serviço da empresa ou de parceiros (terceiros) que mantém convênio com a mesma.

CONTRATO

Representa um acordo entre um cliente e a empresa operadora, referente a um documento, que contém um objeto, direitos e obrigações firmadas entre as partes (contrato), que tem como conseqüência a geração de um item de conta.

Como exemplo podemos ter Contrato de Plano de Expansão, Parcelamento de Conta, Contrato de Terceiro (TV a Cabo, Lista Telefônica), etc.

CONTRATO FATURAMENTO

Representa um acordo entre um cliente e a Empresa Operadora (Contrato), referente ao faturamento, que tem como conseqüência a geração de Itens da Conta.

CONTRATO MEIO ACESSO

Representa um acordo entre um cliente e a empresa operadora (contrato), referente a um meio de acesso, que tem como conseqüência a geração de itens da conta.

CRONOGRAMA

Identifica, para cada composição dos grupos de documentos, as datas de corte para emissão de faturas.

DIVERSOS EM CONTA

Todas as cobranças que a empresa necessita lançar em faturas e que não fazem parte do negócio. Como exemplos podemos ter: restituição de valores por erro em cobrança, prestação de serviços extraordinários, reparos por danos causados à rede, etc.

DOCUMENTO

Documento emitido pela empresa operadora para cobrança ou crédito por meios utilizados ou serviços prestados aos clientes. O documento de recebimento trás em seu corpo todos os itens e serviços utilizados pelo cliente detalhados dentro de um período de faturamento, ou seja, dentro de uma faixa de tempo determinado pela empresa, para que o valor total do documento seja pago pelo cliente e arrecadado pela empresa operadora.

ENDERECO

Contém a estrutura completa de endereços de cobrança utilizados para envio dos documentos de recebimento.

ITEM DA CONTA

Representa um serviço, um encargo, uma comissão ou um desconto associado a um Contrato Meio de Acesso, que após a valoração e apropriação, gera um débito ou um crédito a um cliente.

Como exemplo podemos citar: Ligações interurbanas, assinatura de um telefone fixo, juros e multa por atraso de pagamento, etc.

LANCAMENTO SALDO

Registra todos os lançamentos que implicam em mudança no saldo financeiro de um contrato de faturamento.

Exemplos de lançamentos: Arrecadação, Reclamação de Contas, Agregação, etc.

MANUTENCAO

Contém os dados referentes à cobrança de aparelhagem utilizadas pelo cliente no meio de acesso em que esta instalada, como também a manutenção dos mesmos, através do número de faturamento informado. Estas cobranças serão mensais.

MEIO ACESSO

Contém todos os meios de acesso disponibilizados pela empresa operadora para utilização dos serviços.

Exemplo de meio de acesso: Número de Terminal, Endereço Caixa Postal, Número Cartão Telecard, etc.

OBJETO FATURAMENTO DO PERIODO

Contém os objetos de faturamento que foram selecionados para um determinado período de faturamento.

ORDEM DE SERVICO

Contém todas as ordens de serviço abertas pela empresa operadora.

SERVIÇO MENSAL

Contém os dados referentes ao aluguel ou assinatura do contrato de faturamento e aluguel ou assinatura de aparelhagem que são cobrados mensalmente.

SERVIÇO SUPLEMENTAR

Armazena os registros de cobranças mensais de serviços adicionais. Exemplo: Identificador de Chamadas, Atendimento Simultâneo, etc.

SERVIÇO TERCEIRO

Armazena as cobranças de serviços que não são providos pela empresa operadora. Exemplo: Cobrança de um jornal mensal.

TIPO ITEM EM CONTA

Contém todos os tipos de itens utilizados para o faturamento, tais como serviços (chamada interurbana manual intra, chamada interurbana automática inter a cobrar, etc), encargos (juros, atualização, multa, etc) entre outros.

4.2 Esquema EBV derivado do Sistema de Faturamento.

O esquema de dados relacional, criado para suportar os dados utilizados pelo protótipo do módulo EBV, teve as regras de negócio extraídas do escopo apresentado na seção anterior.

Como o intuito do protótipo é prover acesso rápido das faturas via WEB, precisou-se aumentar o desempenho do esquema original através de uma reestruturação utilizando-se da desnormalização e retirada de informações que seriam irrelevantes ao EBV. A estrutura ainda recebeu algumas características da Orientação a Objetos, pois suporta uma aplicação desenvolvida sob este modelo e necessita persistir objetos na base de dados univocamente.

Devido a estes motivos, teremos então um modelo enxuto em relação ao original e capaz de armazenar objetos que serão identificados por seus OIDs (Object Identifiers), mantendo assim a unicidade necessária.

A Figura 4.2 apresenta o esquema de dados relacional que atualmente suporta o protótipo.

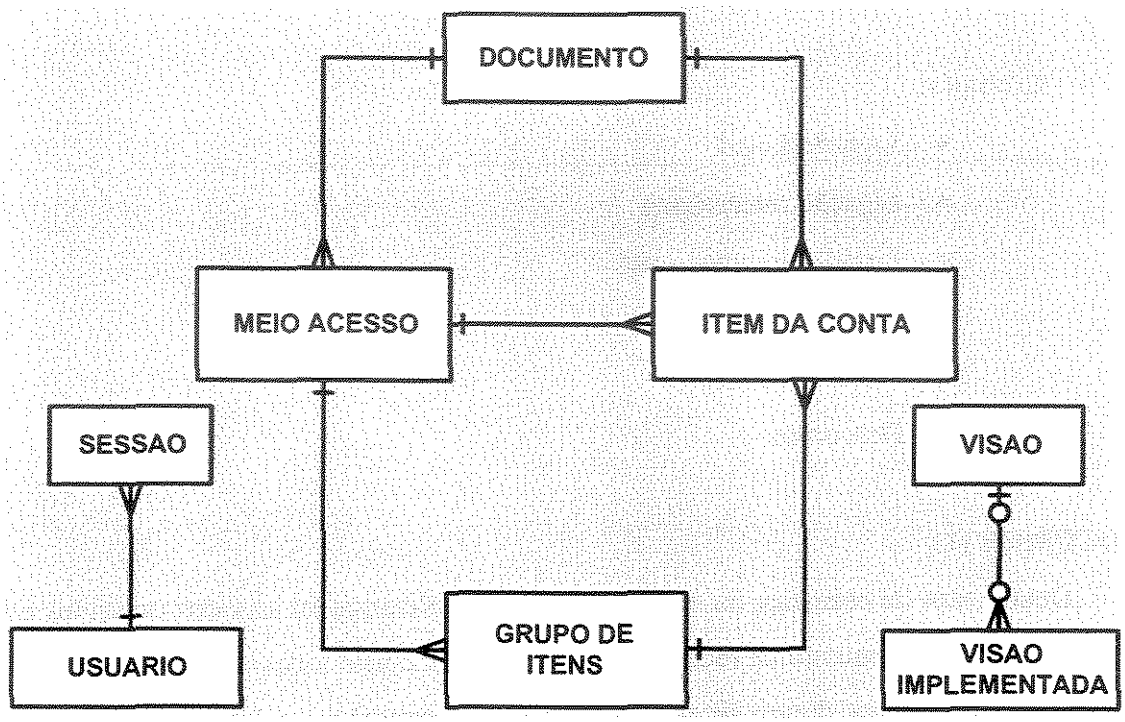

Figura 4.2: Esquema Relacional do módulo EBV

A DDL (Data Definition Language) de criação da base de dados referente ao modelo apresentado na Figura 4.2 encontra-se disponível no Anexo A.

4.2.1 Descrição das Entidades

Apresentaremos neste tópico uma breve descrição de cada entidade contida no esquema demonstrado na Figura 4.2, visando esclarecer alguns pontos do processo de desnormalização realizado sobre o escopo de origem e facilitar a compreensão da estrutura de dados resultante.

GRUPO DE ITENS

Grupos de modalidade de serviço, especificando blocos de serviço, para tratamento diferenciado. Relacionam tipos de itens contidos na conta, pertencentes a uma modalidade de serviço. A conta (documento de cobrança ao cliente) será emitida considerando-se os blocos de serviço definidos, seguindo uma ordem de prioridade para exibição.

Exemplo:

- Ligação Local
- Ligação Interurbana
- Ligação Internacional
- Ligação Celular
- Multa por Atraso
- Descontos

DOCUMENTO

Documento de cobrança ou crédito por meios utilizados ou serviços prestados aos clientes da operadora. O documento de recebimento traz em seu corpo todos os itens e serviços utilizados pelo cliente detalhados dentro de um período de faturamento, ou seja, dentro de uma faixa de tempo determinado pela empresa, para que o valor total do documento seja pago pelo cliente e arrecadado pela empresa operadora.

ITEM DA CONTA

Representa um serviço, um encargo, uma comissão ou um desconto associado a um contrato, que após a valoração e apropriação, gera um débito ou um crédito a um documento.

Exemplos:

- Ligações locais.
- Ligações interurbanas.
- Ligações internacionais.
- Assinatura de um telefone fixo.
- Juros e multa por atraso de pagamento.

A entidade apresenta a estrutura capaz de armazenar todos os itens da conta, portanto se encontra desnormalizada em relação ao esquema de origem, onde os itens da conta se encontram organizados em subtipos.

MEIO ACESSO

Contém todos os dados do cliente e informações do meio de acesso ao qual o cliente tem acesso aos serviços prestados pela mesma.

Exemplos:

- Terminal Fixo
- Terminal Celular
- Terminal Celular Rural

A entidade se encontra desnormalizada por motivos de projeto.

SESSAO

Armazena um registro para cada conecxão que o usuário realizar no sistema.

USUARIO

Armazena informações cadastrais de todos os clientes da empresa operadora que acessarem suas faturas via WEB.

VISAO

Armazena todos os tipos de visões que serão disponibilizadas aos clientes da empresa operadora. Como exemplos podemos citar: visão das contas sumarizadas, visão dos históricos, etc.

VISAO IMPLEMENTADA

Armazena o período de vigência, o nome da classe Java e do componente WEB que implementam cada visão disponibilizada pelo protótipo EBV.

4.3 Resumo

Este capítulo apresentou dois esquemas. O primeiro é parte de uma estrutura complexa que suporta um Sistema de Faturamento, cuja importância foi migrar algumas características necessárias para a construção do esquema de dados que atualmente suporta o protótipo do módulo de visualização de contas EBV. Com este conhecimento, espera-se que seja possível criar um arcabouço que permita a compreensão do próximo capítulo, no qual demonstraremos uma estrutura de dados com as mesmas regras de negócio do esquema relacional, porém utilizando-se das extensões providas pelo paradigma Objeto-Relacional. Com isto pretendemos facilitar a extração dos dados relevantes de nossa base e obter ganho de desempenho na atualização e recuperação dos dados.

Capítulo 5

EBV Objeto-Relacional

A indústria da informática, desde seu início, recria no computador cópias bidimensionais do mundo em que vivemos. Começamos com arquivos, registros, relacionamentos e chegamos aos objetos de hoje[FERN8I].

O objetivo continua o mesmo até hoje: representar de maneira mais fiel possível o mundo que nos cerca. Nossa tarefa, neste capítulo, é empregar algumas melhorias no esquema que atualmente suporta o protótipo do módulo EBV, através de extensões, providas pelo modelo Objeto-Relacional.

Com base na compreensão das informações apresentadas nos capítulos anteriores e o emprego desta nova tecnologia, desenvolvemos um esquema de dados capaz de suportar as mesmas informações que a estrutura relacional que atualmente suporta o protótipo do módulo EBV, mas com novas características que renovam os conceitos de armazenamento de dados.

Este capítulo está organizado da seguinte forma. A seção 5.1 apresenta o esquema objeto desenvolvido neste trabalho. A seção 5.2 demonstra o esquema físico do modelo e os comandos de criação da base de dados. Na seção 5.3 faremos uma discussão sobre os recursos utilizados no desenvolvimento. A seção 5.4 apresenta um resumo do capítulo.

5.1 Esquema Lógico Objeto-Relacional

Empregamos no desenvolvimento desta estrutura, recursos providos pela extensão do modelo Objeto-Relacional e alguns conceitos da Orientação a Objetos como: Classes, Objetos, Atributo, Métodos, Agregação, Encapsulamento, entre outros. Para representar graficamente estes recursos, utilizamos a UML como linguagem de modelagem, pois ela permite comunicar certos conceitos com maior naturalidade do que linguagens alternativas, pois a linguagem natural é muito imprecisa e ela se complica quando se trata de conceitos mais complexos. Outro ponto importante é a padronização que a UML proporciona aos diagramas, pois facilita a integração com várias ferramentas disponíveis no mercado.

O esquema demonstrado na Figura 5.1 apresenta os tipos de objetos que foram utilizados no desenvolvimento do EBV objeto-relacional.

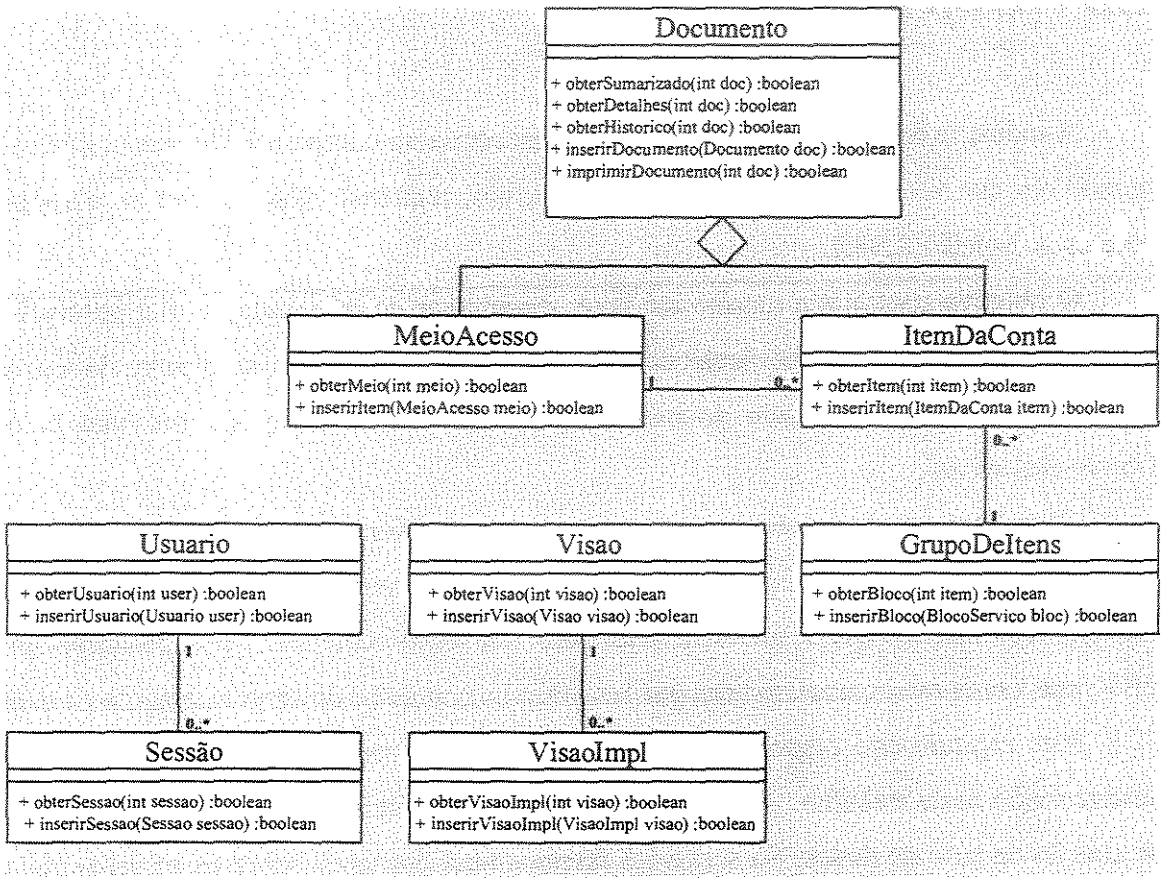

Figura 5.1: Diagrama de Tipos de Objetos

No diagrama os tipos de objetos MeioAcesso e ItemDaConta são agregações presentes em Documento. ItemDaConta possui associações com MeioAcesso e GrupoDeItens. Visão se associa com VisaoImpl, enquanto que Usuario possui uma associação com Sessão.

Em cada um dos tipos de objetos foi criado um método do tipo MAP para garantir que podemos estabelecer ordenação pelo atributo correspondente.

As coleções não foram apresentadas no diagrama. Para que Documento pudesse conter diversos meios de acessos e os itens de faturamento correspondentes, os objetos dos tipos MeioAcesso e ItemDaConta foram organizados em Nested Tables, ou seja, embutidos em Documento como valores em duas colunas [FK00].

5.2 Esquema Físico

A estrutura física elaborada a partir do esquema lógico demonstrado na figura 5.1 apresenta algumas particularidades de implementação que demonstraremos nesta sessão.

No diagrama da Figura 5.1, observa-se que os tipos de objetos MeioAcesso e ItemDaConta são agregações presentes em Documento.

A Figura 5.2 nos mostra como estas agregações foram implementadas no nosso esquema objeto-relacional.

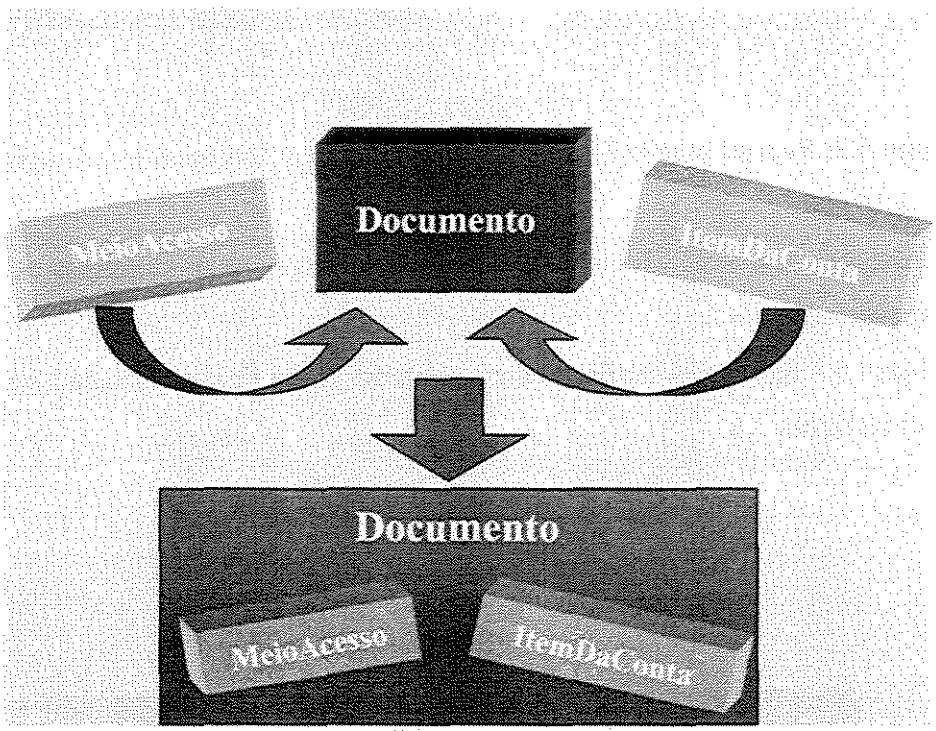

Figura 5.2: Representação Física da Estrutura de Documento

Como podemos ver no diagrama acima, as estruturas que armazenam os meios de acesso e itens da conta, passaram a se integrar de forma aninhada ao objeto Documento. Desta forma passamos a ter um único objeto capaz de armazenar todos os dados de um documento [RH97].

Se observarmos a Figura 5.1, notaremos que existe uma associação entre as classes ItemDaConta e GrupoDeItens. A implementação desta associação ficou simples, como a classe GrupoDeItens vai existir fisicamente, criamos a associação entre os Grupos de itens e o objeto Documento, sendo que os itens da conta agora fazem parte deste objeto.

A figura 5.3 demonstra graficamente a associação entre os itens da conta, que estão embutidos dentro de Documento, e seus respectivos grupos de itens.

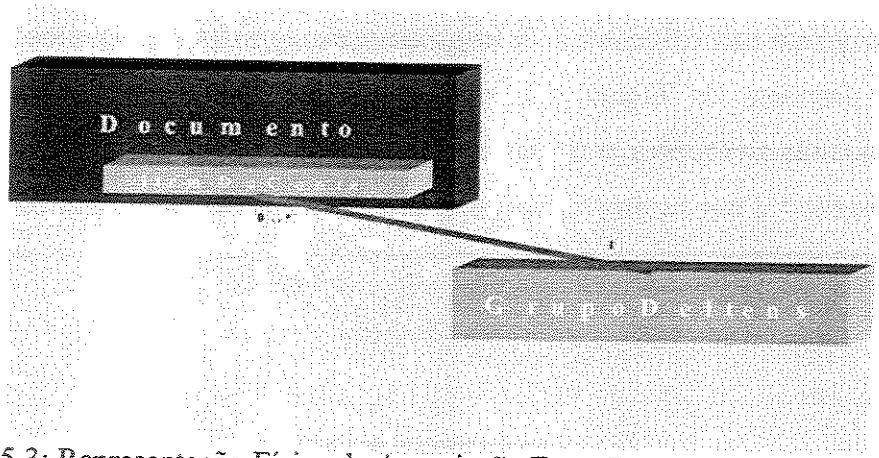

Figura 5.3: Representação Física da Associação Entre Documento e Grupo de Itens

Os objetos Usuario, Sessao, Visao e VisaoImpl foram implementados com suas respectivas associação.

A figura 5.4 representa graficamente a estrutura física final elaborada a partir do esquema demonstrado na figura 5.1.

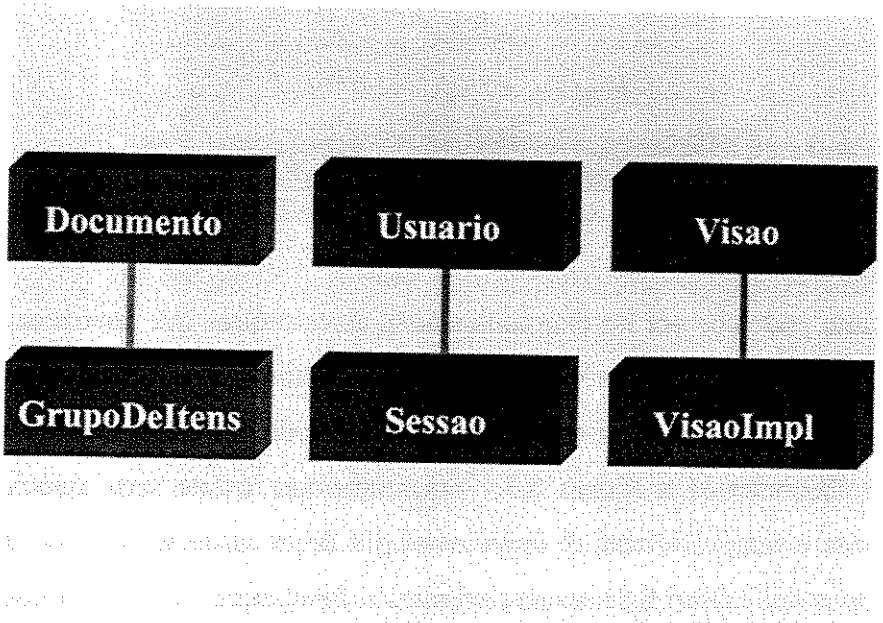

Figura 5.4: Representação gráfica da Estrutura Física

A DDL (*Data Definition Language*) de criação da base de dados referente ao modelo apresentado na figura 5.4 encontra-se disponível no Anexo B.

5.3 Detalhamento dos Recursos Utilizados

Utilizamos conceitos da Orientação a Objetos e recursos disponíveis no modelo Objeto-Relacional para criar uma base de dados mais rica que a relacional apresentada no capítulo anterior.

Em um banco de dados modelado através dos conceitos do modelo Objeto-Relacional, pode-se criar uma tabela com uma coluna cujo tipo seja outra tabela, isto é, tabelas podem ser embutidas em outras tabelas como valores em uma coluna [RH97]. Este recurso de armazenamento leva o nome de *Nested Tables*.

Utilizamos este recurso quando aninhamos meios de acessos e itens das contas dentro de seus respectivos documentos. Podemos dizer então que meios de acessos e itens da contas são agregações presentes em documentos.

O Oracle Server armazena os dados das *Nested Tables* fora das linhas da tabela pai, usando uma propriedade (*store table*) que ficará associada com a coluna da *Nested Table*. A linha pai, na verdade, contém um identificador único associado com a instância correspondente da Nested Table [SK99].

No modelo Relacional, os relacionamentos são restrições que participam das regras de negócio estabelecidas no banco de dados.

O modelo Objeto-Relacional, disponibiliza o recurso Ref (referência) para estabelecer uma ligação com um objeto específico, ou mais formalmente, com uma instância específica de um tipo de objeto.

O uso de referências disponibiliza uma interação transparente entre os objetos persistidos, garantindo ainda acesso mais rápido às informações do que a utilização de junções do modelo Relacional. Utilizamos este recurso nas associações simples entre os tipos de objetos criados, por exemplo, na associação entre ItemDaConta e GrupoDeItens, demonstrada na figura 5.1 [FK00].

Garantimos também a unicidade dos objetos através do uso de Object ID.

Demonstramos aqui o uso de uma pequena parte das potencialidades do modelo Objeto-Relacional.

5.4 Resumo

O capítulo apresentou o esquema objeto-relacional desenvolvido neste trabalho, bem como um detalhamento dos recursos utilizados.

O próximo capítulo apresenta os resultados de uma análise comparativa de desempenho entre os esquemas relacional e objeto-relacional.

Capítulo 6

Análise de Desempenho

Este capítulo demonstra uma análise comparativa de desempenho realizada entre o esquema relacional, que atualmente suporta o protótipo do módulo EBV, e o esquema objeto-relacional elaborado neste trabalho. As estruturas foram implementadas no mesmo ambiente de Hardware e Software e armazenam a mesma massa de dados. Utilizamos algumas ferramentas da Oracle como o TRACE, TKPROF e EXPLAIN PLAN, para nos auxiliar na análise de desempenho de cada esquema.

O capítulo está organizado da seguinte forma. A seção 6.1 apresenta a massa de dados utilizada no processo de análise. A seção 6.2 descreve de forma simplificada a carga de trabalho realizada. A seção 6.3 e 6.4 descrevem respectivamente o ambiente de Hardware e Software. A seção 6.5 descreve os resultados da análise realizada sobre o comparativo dos esquemas apresentados. Finalmente, a seção 6.6 resume o capítulo.

6.1 Descrição dos Dados Utilizados

Os dados utilizados nos testes de desempenho foram retirados da base de testes do protótipo EBV.

A base conta com o seguinte volume de dados:

Dados	Quantidade
Documentos	5072
Meio acesso	15924
Itens da conta	616497
Grupo de itens	42144
Usuários	128
Sessões	233
Visões	5
Visões Implementadas	5

Tabela 6.1: Volume de dados na base para teste

6.2 Carga de Trabalho

A carga de trabalho demonstrada nesta sessão foi elaborada com base nas principais consultas disponíveis no módulo de visualização de contas.

Os códigos referentes a cada caso demonstrado abaixo se encontram disponíveis no Anexo C.

1) Recuperar informações de cadastro de usuários para simples conferência, através do número do meio de acesso (Telefone).

2) Recuperar informações de notas fiscais de determinado cliente, por meio do CPF ou NRC que é uma identificação específica do sistema de faturamento.

3) Recuperar para consulta o número da nota fiscal de um meio de acesso específico.

4) Selecionar informações do documento mais recente para determinado cliente. Podemos utilizar como chaves de busca o CPF ou NRC.

5) Selecionar informações do documento mais recente para determinado meio de acesso.

6) Sumarizar as informações de chamadas telefônicas agrupadas por tarifa.

Dentre estas informações teremos: Totais de chamadas, duração total e valores líquidos e brutos.

7) Obter o número da nota fiscal de cada documento emitido para um meio de acesso.

8) Recuperar valores líquidos e brutos de todas as chamadas de um meio de acesso em um determinado período.

9) Recuperar o agregador, ou seja, o meio de acesso responsável pelas chamadas de cada documento.

10)Selecionar detalhes de todos os itens de faturamento de um determinado documento de recebimento de um meio de acesso.

11) Detalhar a totalização por bloco de serviço de um meio de acesso em um documento.

12) Detalhamento dos totais gerais de um determinado documento emitido para um meio de acesso.

13) Detalhar o total geral de Chamadas de um determinado documento.

14) Recuperar o agregador do meio de acesso passado como filtro.

15) Detalhar os totais do documento agrupando por tipo item faturamento.

16) Alteração no valor líquido de um item da conta.

17) Carga de dados e atualização de índices.

6.3 Descrição do ambiente de Hardware

Máquina: Sun E4500
Processadores: 8 de 400MHz cada, totalizando 3200MHz
Discos: 4 discos internos de 9Gb + Array Sun com 2 controladoras (c2 e c3)
Distribuição dos discos:

Controladora c2 da salmao- ligado no Array A3500

c2t4d1s2	Sd118	/l/disk1
c2t4d3s2	Sd120	
c2t4d5s2	Sd122	/l/disk7
c2t4d7s2	Sd124	/l/disk8 /l/rep1 – binario do Oracle (Home da conta)
c2t4d9s2	Sd753	
c2t4d11s2	Sd755	
c2t5d0s2	Sd34	/l/home0 – cold_backup
c2t5d2s2	Sd329	/l/home0
c2t5d4s2	Sd331	
c2t5d6s2	Sd333	
c2t5d8s2	Sd864	/l/home1 – dbf2
c2t5d10s2	Sd866	/l/disk3

Tabela 6.2: Distribuição dos discos na controladora c2

Controladora c3 da salmao - ligado no array A3500

c3t4d1s2	Sd328	/l/home0
c3t4d3s2	Sd330	/l/home0
c3t4d5s2	Sd332	
c3t4d7s2	Sd334	/l/home2
c3t4d9s2	Sd865	/l/disk2
c3t4d11s2	Sd867	/l/disk6 – dbf1
c3t5d0s2	Sd4	/l/disk0 – oradata/salmao817
c3t5d2s2	Sd119	
c3t5d4s2	sd121	
c3t5d6s2	sd123	
c3t5d8s2	sd752	
c3t5d10s2	sd754	

Tabela 6.3: Distribuição dos discos na controladora c3

6.4 Descrição do ambiente de Software

Solaris 2.6
Oracle 8.1.7.0.0

Descrição dos parâmetros iniciais do oracle

Parametro	Descrição	Valor inicial
Tablespace RBS	área de rollback	1GB
Segmentos de rollback	área de rollback	16 segmentos de rollback com initial e next de 1MB, minexentes 20, optimal 50MB
Tablespace TEMP	Área de sort em disco	2GB
Tablespace SYSTEM	área de objetos do Oracle – dicionário de dados	275MB
Redo logfiles	Arquivos de recovery da instância no caso de crash – armazena todas as alterações	3 redo logfiles de 100MB cada, todos em mesmo disco
Db_block_size	bloco Oracle	8KB
Database files	Arquivos do banco	os datafiles, redo logfiles e control files estão todos no mesmo disco com mesma controladora
Db_block_buffers	Buffer para carga de blocos de dados, unidade em blocos Oracle	= 54931
Shared_pool_size	Área que contém sentenças SQL, códigos parseados (compilados), plano de execução, cache do dicionário de dados, shared cursors, stored procedures, estruturas de controle	= 450000000
Log_checkpoint_interval	Especifica a freqüência de checkpoint em termos do número de blocos que podem existir entre um checkpoint e o último bloco escrito no redo. Este número é em blocos do SO e não do Oracle. Apesar deste, o checkpoint sempre ocorre a cada log switch. Checkpoint influencia o tempo de instance recovery. Checkpoints muito freqüentes podem influenciar a performance do banco.	= 50000

Log_checkpoint_timeout	Especifica que o checkpoint deve ocorrer na mesma posição onde a última escrita ao redo estava há este número de segundos atrás. Setando para zero desabilita checkpoints baseeados em tempo.	= 0
Processes	Número máximo de processos	= 200
Log_buffer	Buffer do redo logfile, antes de escrever no redo logfile as informações sobre alterações ao banco são escritas nesta.	= 10240000
Timed_statistics	Seta estatística de tempo	= true
Sort_area_size	Área de sort em memória, por sessão.	= 4096000
Sort_area_retained_size	Espaço da área de sort que será mantida (não desalocada) após sort	= 1024000
Optimizer_mode	Otimizador baseado em regras fixas que possuem um ranking de custo. Outra opção é o otimizador baseado em estatísticas	= rule
Pre_page_sga	Quando setado, pre-aloca no momento do startup do Oracle, toda a SGA, do contrário a memória vai crescendo conforme necessidade em segmentos de memória.	= true
Db_writer_processes	Processos que escrevem alterações aos blocos Oracle nos datafiles correspondentes	= 4

Tabela 6.4: Parâmetros iniciais Oracle

6.5 Análise

Demonstraremos uma análise comparativa de desempenho realizada entre o esquema relacional, que atualmente suporta o protótipo do módulo EBV, e o esquema objeto-relacional elaborado neste trabalho.

A carga de trabalho foi enfocada nas principais funções do protótipo EBV, visando assim à recuperação das informações armazenadas na base de dados.

Os testes foram realizados diversas vezes sobre um servidor exclusivo, os resultados apresentados são médias retiradas de cinco execuções para cada caso de teste em condições semelhantes de memória, a fim de reduzir a margem de erro das métricas coletadas.

Utilizamos três ferramentas para nos auxiliar nesta análise [BIND00,KYTE8I]. O EXPLAIN PLAN possibilitou a visualização do plano de execução utilizado pelo otimizador do SGBD Oracle para cada instrução SQL testada. Utilizamos o TRACE e o visualizador TKPROF, fornecidos pela Oracle, para extrair métricas que possibilitaram uma análise comparativa mais detalhada.

Apresentamos abaixo as métricas utilizadas na realização de nossa análise de desempenho:

Count	Número de vezes que o procedimento foi executado.
cpu	Tempo em que a CPU esta executando.
Elapsed	Tempo total de execução.
disk	Número de leitura físicas no disco (E/S).
Query	Número de buffers alocados para consistir a leitura.
Current	Número de buffers alocados no modo atual (normalmente para update)
rows	Número de linhas retornadas.

Tabela 6.5: Métricas utilizadas para teste

6.5.1 Teste 1

Objetivo

Recuperar informações de usuários para simples conferência, através do número do meio de acesso (Telefone).

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
TABLE ACCESS BY INDEX ROWID          PRPERF1              T_USUARIO
INDEX RANGE SCAN                     PRPERF1              IDX_USER_R1
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.02	0.00	1	0	0	0
Fetch	1	0.03	0.06	1	2	0	1
Total	3	0.05	0.06	2	2	0	1

Tabela 6.6: Valores obtidos para o esquema relacional no teste 1

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
TABLE ACCESS BY INDEX ROWID          PRPERF1              OT_USUARIO
INDEX RANGE SCAN                     PRPERF1              IDX_USER1
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.01	0.01	1	0	0	0
Fetch	1	0.03	0.05	1	2	0	1
Total	3	0.04	0.06	2	2	0	1

Tabela 6.7: Valores obtidos para o esquema objeto-relacional no teste 1

Conclusão

As consultas apresentaram praticamente o mesmo plano de execução e os mesmos resultados na análise. Embora o tabela Objeto apresente algumas diferenças, pois foi criada a

partir de um tipo definido pelo usuário, a estrutura interna das tabelas são praticamente as mesmas. Com isto, não tivemos diferenças nas métricas da análise.

6.5.2 Teste 2

Objetivo

Recuperar informações de notas fiscais de determinado cliente, utilizando o CPF como chave de busca.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
  SORT UNIQUE
   NESTED LOOPS
    NESTED LOOPS
     NESTED LOOPS
      TABLE ACCESS BY INDEX ROWID          PRPERF1          T_MEIO_ACESSO
        INDEX RANGE SCAN                   PRPERF1          IDX_MEIO_R1
        INDEX UNIQUE SCAN                  PRPERF1          PK_T_DOCUMENTO
      TABLE ACCESS BY INDEX ROWID          PRPERF1          T_MEIO_ACESSO
        INDEX RANGE SCAN                   PRPERF1          IDX_MEIO_R2
      TABLE ACCESS BY INDEX ROWID          PRPERF1          T_DOCUMENTO
        INDEX UNIQUE SCAN                  PRPERF1          PK_T_DOCUMENTO
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	1	0	0	0
Fetch	1	0.16	0.17	1	11080	0	8
Total	3	0.16	0.17	2	11080	0	8

Tabela 6.8: Valores obtidos para o esquema relacional no teste 2

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
  SORT UNIQUE
   NESTED LOOPS
    NESTED LOOPS
     NESTED LOOPS
      TABLE ACCESS BY INDEX ROWID          PRPERF1              OTN_MEIO_ACESSO
          INDEX RANGE SCAN                 PRPERF1              IDX_MEIO4
      TABLE ACCESS BY INDEX ROWID          PRPERF1              OT_DOCUMENTO
          INDEX UNIQUE SCAN                PRPERF1              SYS_C0019524
      TABLE ACCESS BY INDEX ROWID          PRPERF1              OT_DOCUMENTO
          INDEX UNIQUE SCAN                PRPERF1              SYS_C0019522
      TABLE ACCESS BY INDEX ROWID          PRPERF1              OTN_MEIO_ACESSO
          INDEX RANGE SCAN                 PRPERF1              IDX_MEIO3
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	1	0	0	0
Fetch	1	0.03	0.03	1	1770	0	8
Total	3	0.03	0.03	2	1770	0	8

Tabela 6.9: Valores obtidos para o esquema objeto-relacional no teste 2

Conclusão

Os dois esquemas apresentaram planos de execução semelhantes mas a estrutura aninhada do esquema objeto-relacional propicia um menor tempo de CPU e melhor utilização de memória em relação ao esquema relacional dadas a forma de alocação das linhas na tabela aninhada e a redução das estruturas de índices, respectivamente.

6.5.3 Teste 3

Objetivo

Recuperar para consulta informações da nota fiscal de um meio de acesso específico.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
  SORT UNIQUE
   NESTED LOOPS
    NESTED LOOPS
     NESTED LOOPS
      TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
          INDEX RANGE SCAN                 PRPERF1      IDX_MEIO_R3
          INDEX UNIQUE SCAN                PRPERF1      PK_T_DOCUMENTO
      TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
          INDEX RANGE SCAN                 PRPERF1      IDX_MEIO_R2
      TABLE ACCESS BY INDEX ROWID          PRPERF1      T_DOCUMENTO
          INDEX UNIQUE SCAN                PRPERF1      PK_T_DOCUMENTO
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.02	0	0	0	0
Execute	1	0.03	0.01	0	0	0	0
Fetch	1	0.02	0.02	0	60	0	4
Total	3	0.05	0.05	0	60	0	4

Tabela 6.10: Valores obtidos para o esquema relacional no teste 3

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
```

```
SORT UNIQUE
 NESTED LOOPS
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                          PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                         PRPERF1      SYS_C0019524
   TABLE ACCESS BY INDEX ROWID                 PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                         PRPERF1      SYS_C0019522
  TABLE ACCESS BY INDEX ROWID                  PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                          PRPERF1      IDX_MEIO3
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.01	0	0	0	0
Execute	1	0.01	0.01	0	0	0	0
Fetch	1	0.02	0.03	0	54	0	4
Total	3	0.03	0.05	0	54	0	4

Tabela 6.11: Valores obtidos para o esquema objeto-relacional no teste 3

Conclusão

Os esquemas apresentaram o mesmo tempo total de execução mas obtivemos algumas diferenças nas outras métricas. O esquema objeto-relacional apresentou um melhor desempenho na fase de Parse, devido a agilidade na recuperação das linhas alocadas na tabela aninhada, mas perdeu na realização do Fetch devido a reorganização dos dados.

6.5.4 Teste 4

Objetivo

Selecionar informações do documento mais recente para determinado cliente utilizando o CPF/CGC como chave de busca.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 FILTER
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID                PRPERF1      T_MEIO_ACESSO
    INDEX RANGE SCAN                          PRPERF1      IDX_MEIO_R1
   TABLE ACCESS BY INDEX ROWID                PRPERF1      T_DOCUMENTO
    INDEX UNIQUE SCAN                         PRPERF1      PK_T_DOCUMENTO
  SORT AGGREGATE
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID               PRPERF1      T_MEIO_ACESSO
     INDEX RANGE SCAN                         PRPERF1      IDX_MEIO_R1
    TABLE ACCESS BY INDEX ROWID               PRPERF1      T_DOCUMENTO
     INDEX UNIQUE SCAN                        PRPERF1      PK_T_DOCUMENTO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.01	0	0	0	0
Fetch	1	0.01	0.01	0	252	0	10
Total	3	0.01	0.02	0	252	0	10

Tabela 6.12: Valores obtidos para o esquema relacional no teste 4

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 FILTER
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_MEIO_ACESSO
    INDEX RANGE SCAN                          PRPERF1      IDX_MEIO4
   TABLE ACCESS BY INDEX ROWID                PRPERF1      OT_DOCUMENTO
    INDEX UNIQUE SCAN                         PRPERF1      SYS_C0019524
  SORT AGGREGATE
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID               PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                         PRPERF1      IDX_MEIO4
    TABLE ACCESS BY INDEX ROWID               PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                        PRPERF1      SYS_C0019524
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.01	0	256	0	10
Total	3	0.01	0.01	0	256	0	10

Tabela 6.13: Valores obtidos para o esquema objeto-relacional no teste 4

Conclusão

Os dois esquemas apresentaram um bom plano de execução e tiveram o mesmo tempo de CPU, mas o objeto-relacional apresentou uma pequena redução no tempo total de execução.

6.5.5 Teste 5

Objetivo

Selecionar informações do documento mais recente para determinado meio de acesso.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 FILTER
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID              PRPERF1      T_MEIO_ACESSO
    INDEX RANGE SCAN                        PRPERF1      IDX_MEIO_R3
   TABLE ACCESS BY INDEX ROWID              PRPERF1      T_DOCUMENTO
    INDEX UNIQUE SCAN                       PRPERF1      PK_T_DOCUMENTO
  SORT AGGREGATE
```

```
NESTED LOOPS
 TABLE ACCESS BY INDEX ROWID            PRPERF1      T_MEIO_ACESSO
  INDEX RANGE SCAN                      PRPERF1      IDX_MEIO_R3
 TABLE ACCESS BY INDEX ROWID            PRPERF1      T_DOCUMENTO
  INDEX UNIQUE SCAN                     PRPERF1      PK_T_DOCUMENTO
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.10	0.13	0	0	0	0
Execute	1	0.00	0.02	0	0	0	0
Fetch	1	0.42	0.56	2	15920	0	1
Total	3	0.52	0.71	2	15920	0	1

Tabela 6.14: Valores obtidos para o esquema relacional no teste 5

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 FILTER
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID          PRPERF1      OTN_MEIO_ACESSO
    INDEX RANGE SCAN                    PRPERF1      IDX_MEIO5
   TABLE ACCESS BY INDEX ROWID          PRPERF1      OT_DOCUMENTO
    INDEX UNIQUE SCAN                   PRPERF1      SYS_C0019524
  SORT AGGREGATE
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID         PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                   PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID         PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                  PRPERF1      SYS_C0019524
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.05	0	0	0	0
Execute	1	0.01	0.00	0	0	0	0
Fetch	1	0.07	0.08	1	345	0	1
total	3	0.09	0.13	1	345	0	1

Tabela 6.15: Valores obtidos para o esquema objeto-relacional no teste 5

Conclusão

Obtivemos bons planos de execução nos dois esquemas mas o esquema objeto-relacional apresentou um desempenho superior. Obteve significativas diferenças em tempo de CPU, utilização de memória, I/O a disco e tempo total de execução. Com isto, podemos perceber que o join realizado internamente na estrutura objeto-relacional é mais eficiente que o realizado no esquema relacional pois recupera as informações relacionadas com um menor custo de recursos.

6.5.6 Teste 6

Objetivo

Sumarizar as chamadas telefônicas de uma nota fiscal agrupadas por tarifa. Dentre estas informações teremos: Totais de chamadas, duração total e valores líquidos e brutos.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
  SELECT STATEMENT, GOAL = RULE
SORT GROUP BY
 NESTED LOOPS
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID          PRPERF1      T_DOCUMENTO
    INDEX RANGE SCAN                    PRPERF1      IDX_DOC_R2
   TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
    INDEX RANGE SCAN                    PRPERF1      IDX_MEIO_R2
  TABLE ACCESS BY INDEX ROWID           PRPERF1      T_ITEM_FATURAMENTO
   AND-EQUAL
    INDEX RANGE SCAN                    PRPERF1      IDX_FK_MEIO_ACESSO
    INDEX RANGE SCAN                    PRPERF1      IDX_FK_DOC_ITEM
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	2	0.00	0.00	0	0	0	0
Execute	2	0.02	0.02	0	0	0	0
Fetch	2	0.06	0.12	6	500	0	2
total	6	0.08	0.14	6	500	0	2

Tabela 6.16: Valores obtidos para o esquema relacional no teste 6

Esquema Objeto-Relacional

Plano de Execução

```
  SELECT STATEMENT, GOAL = RULE
SORT GROUP BY
 NESTED LOOPS
  TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_ITEM_FATURAMENTO
   INDEX RANGE SCAN                          PRPERF1      IDX_ITEM2
  TABLE ACCESS BY INDEX ROWID                PRPERF1      OT_DOCUMENTO
   INDEX UNIQUE SCAN                         PRPERF1      SYS_C0019523
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	2	0.00	0.00	0	0	0	0
Execute	2	0.01	0.02	0	0	0	0
Fetch	2	0.03	0.03	2	20	0	2
total	6	0.04	0.05	2	20	0	2

Tabela 6.17: Valores obtidos para o esquema objeto-relacional no teste 6

Conclusão

Obtivemos menor tempo de CPU, I/O a disco e tempo de execução. Isto se deve a estrutura aninhada implementada no esquema objeto-relacional.

6.5.7 Teste 7

Objetivo

Obter o número da nota fiscal de cada documento emitido para um meio de acesso.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS FULL                    PRPERF1     T_DOCUMENTO
     TABLE ACCESS BY INDEX ROWID          PRPERF1     T_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1     IDX_MEIO_R2
    TABLE ACCESS BY INDEX ROWID           PRPERF1     T_MEIO_ACESSO
     INDEX RANGE SCAN                     PRPERF1     IDX_MEIO_R3
   TABLE ACCESS BY INDEX ROWID            PRPERF1     T_DOCUMENTO
    INDEX UNIQUE SCAN                     PRPERF1     PK_T_DOCUMENTO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	1.27	1.40	1	76638	7	3
Total	3	1.27	1.41	1	76638	7	3

Tabela 6.18: Valores obtidos para o esquema relacional no teste 7

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID          PRPERF1      OTN_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_MEIO5
     TABLE ACCESS BY INDEX ROWID          PRPERF1      OT_DOCUMENTO
      INDEX UNIQUE SCAN                   PRPERF1      SYS_C0019524
    TABLE ACCESS BY INDEX ROWID           PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                     PRPERF1      IDX_MEIO3
   TABLE ACCESS BY INDEX ROWID            PRPERF1      OT_DOCUMENTO
    INDEX UNIQUE SCAN                     PRPERF1      SYS_C0019524
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.00	0.01	2	38	0	3
total	3	0.01	0.02	2	38	0	3

Tabela 6.19: Valores obtidos para o esquema objeto-relacional no teste 7

Conclusão

Não foi possível otimizar mais o plano de execução aplicado ao esquema relacional e por isto obtivemos um acesso *full* na tabela de documentos. O esquema objeto-relacional apresentou um significativo ganho de desempenho sobre seu concorrente apresentando um ótimo plano de execução.

6.5.8 Teste 8

Objetivo

Recuperar valores líquidos e brutos de todas as chamadas de um meio de acesso em um determinado período.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID          PRPERF1      T_DOCUMENTO
     INDEX RANGE SCAN                    PRPERF1      IDX_DOC_R2
    TABLE ACCESS BY INDEX ROWID          PRPERF1      T_ITEM_FATURAMENTO
     INDEX RANGE SCAN                    PRPERF1      IDX_FK_DOC_ITEM
   TABLE ACCESS BY INDEX ROWID           PRPERF1      T_MEIO_ACESSO
    INDEX UNIQUE SCAN                    PRPERF1      PK_T_M_ACESSO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.04	0.10	6	250	7	1
total	3	0.04	0.10	6	250	7	1

Tabela 6.20: Valores obtidos para o esquema relacional no teste 8

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
SORT GROUP BY
 NESTED LOOPS
  TABLE ACCESS BY INDEX ROWID            PRPERF1      OT_DOCUMENTO
   INDEX RANGE SCAN                      PRPERF1      IDX_DOC
  TABLE ACCESS BY INDEX ROWID            PRPERF1      OTN_ITEM_FATURAMENTO
   INDEX RANGE SCAN                      PRPERF1      IDX_ITEM
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.03	5	10	0	1
total	3	0.01	0.03	5	10	0	1

Tabela 6.21: Valores obtidos para o esquema objeto-relacional no teste 8

Conclusão

As consultas apresentaram bons planos de execução. O esquema objeto-relacional mais uma vez apresentou menor tempo de CPU e elapsed, conseguindo ainda, menos acesso a disco.

6.5.9 Teste 9

Objetivo

Recuperar informações do agregador, ou seja, o meio de acesso responsável pelas chamadas de cada documento.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS FULL                    PRPERF1      T_DOCUMENTO
     TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_MEIO_R2
    TABLE ACCESS BY INDEX ROWID           PRPERF1      T_MEIO_ACESSO
```

```
  INDEX RANGE SCAN                          PRPERF1      IDX_MEIO_R3
TABLE ACCESS BY INDEX ROWID                 PRPERF1      T_DOCUMENTO
 INDEX UNIQUE SCAN                          PRPERF1      PK_T_DOCUMENTO
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.02	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	1.70	1.71	6	101973	7	3
Total	3	1.71	1.73	6	101973	7	3

Tabela 6.22: Valores obtidos para o esquema relacional no teste 9

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID       PRPERF1      OTN_MEIO_ACESSO
      INDEX RANGE SCAN                 PRPERF1      IDX_MEIO5
     TABLE ACCESS BY INDEX ROWID       PRPERF1      OT_DOCUMENTO
      INDEX UNIQUE SCAN                PRPERF1      SYS_C0019524
    TABLE ACCESS BY INDEX ROWID        PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                  PRPERF1      IDX_MEIO3
   TABLE ACCESS BY INDEX ROWID         PRPERF1      OT_DOCUMENTO
    INDEX UNIQUE SCAN                  PRPERF1      SYS_C0019524
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.03	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.02	2	56	0	3
total	3	0.01	0.05	2	56	0	3

Tabela 6.23: Valores obtidos para o esquema objeto-relacional no teste 9

Conclusão

O plano de execução obtido pela consulta relacional é inferior ao da consulta objeto-relacional pois não foi permitido indexar todas as tabelas utilizadas, impactando assim no tempo de elapsed.

O esquema objeto-relacional apresentou menor tempo de CPU e acesso a disco devido a estrutura aninhada das tabelas.

6.5.10 Teste 10

Objetivo

Selecionar detalhes de todos os itens da conta de um determinado documento de recebimento de um meio de acesso.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID          PRPERF1      T_DOCUMENTO
      INDEX RANGE SCAN                    PRPERF1      IDX_DOC_R2
     TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_MEIO_R3
    TABLE ACCESS BY INDEX ROWID           PRPERF1      T_ITEM_FATURAMENTO
     AND-EQUAL
      INDEX RANGE SCAN                    PRPERF1      IDX_FK_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_FK_DOC_ITEM
   TABLE ACCESS BY INDEX ROWID            PRPERF1      T_BLOCO_SERVICO
    INDEX UNIQUE SCAN                     PRPERF1      PK_T_BLOCO_SERVICO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.03	0.11	11	117	0	30
total	3	0.04	0.12	11	117	0	30

Tabela 6.24: Valores obtidos para o esquema relacional no teste 10

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT ORDER BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID          PRPERF1       OTN_MEIO_ACESSO
     INDEX RANGE SCAN                    PRPERF1       IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID          PRPERF1       OT_DOCUMENTO
     INDEX UNIQUE SCAN                   PRPERF1       SYS_C0019524
   TABLE ACCESS BY INDEX ROWID           PRPERF1 OTN_ITEM_FATURAMENTO
     INDEX RANGE SCAN                    PRPERF1       IDX_ITEM_FK
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.01	0	0	0	0
Fetch	1	0.00	0.03	4	32	0	30
Total	3	0.01	0.05	4	32	0	30

Tabela 6.25: Valores obtidos para o esquema objeto-relacional no teste 10

Conclusão

Neste teste todo plano de execução se apresenta indexado mas o esquema objeto-relacional se demonstra superior pelo menor acesso a disco e melhor tempo de processamento. O desempenho apresentado pelo esquema objeto se deve a estrutura aninhada que se apresenta em Documento, pois quando solicitamos um item do Documento, todos os itens são retornados de uma só vez, reduzindo assim o acesso ao disco.

6.5.11 Teste 11

Objetivo

Detalhar um documento de um meio de acesso, totalizando por bloco de serviço.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID          PRPERF1      T_DOCUMENTO
     INDEX RANGE SCAN                    PRPERF1      IDX_DOC_R2
    TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
     INDEX RANGE SCAN                    PRPERF1      IDX_MEIO_R3
   TABLE ACCESS BY INDEX ROWID           PRPERF1     T_ITEM_FATURAMENTO
    AND-EQUAL
     INDEX RANGE SCAN                    PRPERF1      IDX_FK_MEIO_ACESSO
     INDEX RANGE SCAN                    PRPERF1      IDX_FK_DOC_ITEM
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.10	8	28	0	2
Total	3	0.02	0.11	8	28	0	2

Tabela 6.26: Valores obtidos para o esquema relacional no teste 11

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                          PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                         PRPERF1      SYS_C0019524
   TABLE ACCESS BY INDEX ROWID                 PRPERF1    OTN_ITEM_FATURAMENTO
    INDEX RANGE SCAN                           PRPERF1      IDX_ITEM_FK
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.03	3	28	0	2
Total	3	0.02	0.04	3	28	0	2

Tabela 6.27: Valores obtidos para o esquema objeto-relacional no teste 11

Conclusão

Podemos notar que a estrutura que armazena documentos no esquema objeto-relacional apresenta maior desempenho do que no relacional. Na maioria dos casos temos menor tempo de CPU e elapsed. O acesso a disco também é menor devido ao modo em que os documentos e seus itens são armazenados.

6.5.12 Teste 12

Objetivo

Detalhamento dos totais gerais de um determinado documento de um meio de acesso.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID            PRPERF1      T_DOCUMENTO
      INDEX RANGE SCAN                      PRPERF1      IDX_DOC_R2
     TABLE ACCESS BY INDEX ROWID            PRPERF1      T_MEIO_ACESSO
      INDEX RANGE SCAN                      PRPERF1      IDX_MEIO_R3
    TABLE ACCESS BY INDEX ROWID             PRPERF1  T_ITEM_FATURAMENTO
     AND-EQUAL
      INDEX RANGE SCAN                      PRPERF1     IDX_FK_MEIO_ACESSO
      INDEX RANGE SCAN                      PRPERF1     IDX_FK_DOC_ITEM
   TABLE ACCESS BY INDEX ROWID              PRPERF1     T_BLOCO_SERVICO
    INDEX UNIQUE SCAN                       PRPERF1     PK_T_BLOCO_SERVICO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.08	9	206	0	1
total	3	0.01	0.08	9	206	0	1

Tabela 6.28: Valores obtidos para o esquema relacional no teste 12

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID             PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                       PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID             PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                      PRPERF1      SYS_C0019524
   TABLE ACCESS BY INDEX ROWID              PRPERF1      OTN_ITEM_FATURAMENTO

    INDEX RANGE SCAN                        PRPERF1      IDX_ITEM_FK
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.02	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.02	0.04	4	32	0	1
total	3	0.03	0.06	4	32	0	1

Tabela 6.29: Valores obtidos para o esquema objeto-relacional no teste 12

Conclusão

Tivemos um bom plano de execução para os dois esquemas mas o esquema objeto-relacional se sobressai sobre o relacional, apresentando menor tempo de execução e menos acesso a disco. Neste teste também vimos a utilização da referência ao invés da chave estrangeira. Este recurso tem o objetivo de acesso direto e rápido, sendo superior aos relacionamentos do modelo Relacional.

6.5.13 Teste 13

Objetivo

Detalhar o total geral de Chamadas de um determinado documento, agrupando pelo período das chamadas.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
```

```
NESTED LOOPS
 NESTED LOOPS
  NESTED LOOPS
   TABLE ACCESS BY INDEX ROWID                PRPERF1      T_DOCUMENTO
    INDEX RANGE SCAN                          PRPERF1      IDX_DOC_R2
   TABLE ACCESS BY INDEX ROWID                PRPERF1      T_MEIO_ACESSO
    INDEX RANGE SCAN                          PRPERF1      IDX_MEIO_R3
  TABLE ACCESS BY INDEX ROWID                 PRPERF1      T_ITEM_FATURAMENTO
   AND-EQUAL
    INDEX RANGE SCAN                          PRPERF1      IDX_FK_MEIO_ACESSO
    INDEX RANGE SCAN                          PRPERF1      IDX_FK_DOC_ITEM
 TABLE ACCESS BY INDEX ROWID                  PRPERF1      T_BLOCO_SERVICO
  INDEX UNIQUE SCAN                           PRPERF1      PK_T_BLOCO_SERVICO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.06	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.02	0.06	2	27	0	1
Total	3	0.03	0.12	2	27	0	1

Tabela 6.30: Valores obtidos para o esquema relacional no teste 13

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID               PRPERF1      OTN_MEIO_ACESSO
     INDEX RANGE SCAN                         PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID               PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                        PRPERF1      SYS_C0019524
   TABLE ACCESS BY INDEX ROWID                PRPERF1    OTN_ITEM_FATURAMENTO
    INDEX RANGE SCAN                          PRPERF1      IDX_ITEM_FK
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.02	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.00	0.05	1	42	0	1
Total	3	0.01	0.07	1	42	0	1

Tabela 6.31: Valores obtidos para o esquema objeto-relacional no teste 13

Conclusão

O esquema objeto-relacional apresentou menor acesso a disco e melhor tempo de execução na sumarização das chamadas graças a estrutura aninhada e o uso de referências (REF).

6.5.14 Teste 14

Objetivo

Recuperar o agregador do meio de acesso passado como chave de busca.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID          PRPERF1      T_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_MEIO_R3
     INDEX UNIQUE SCAN                    PRPERF1      PK_T_DOCUMENTO
    TABLE ACCESS BY INDEX ROWID           PRPERF1      T_MEIO_ACESSO
     INDEX RANGE SCAN                     PRPERF1      IDX_MEIO_R2
   TABLE ACCESS BY INDEX ROWID            PRPERF1      T_DOCUMENTO
    INDEX UNIQUE SCAN                     PRPERF1      PK_T_DOCUMENTO
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.03	0.03	2	20	0	1
total	3	0.04	0.04	2	20	0	1

Tabela 6.32: Valores obtidos para o esquema relacional no teste 14

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT UNIQUE
  NESTED LOOPS
   NESTED LOOPS
    NESTED LOOPS
     TABLE ACCESS BY INDEX ROWID          PRPERF1      OTN_MEIO_ACESSO
      INDEX RANGE SCAN                    PRPERF1      IDX_MEIO5
     TABLE ACCESS BY INDEX ROWID          PRPERF1      OT_DOCUMENTO
      INDEX UNIQUE SCAN                   PRPERF1      SYS_C0019524
    TABLE ACCESS BY INDEX ROWID           PRPERF1      OT_DOCUMENTO
     INDEX UNIQUE SCAN                    PRPERF1      SYS_C0019522
   TABLE ACCESS BY INDEX ROWID            PRPERF1      OTN_MEIO_ACESSO
    INDEX RANGE SCAN                      PRPERF1      IDX_MEIO3
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.00	0.03	1	14	0	1
Total	3	0.01	0.04	1	14	0	1

Tabela 6.33: Valores obtidos para o esquema objeto-relacional no teste 14

Conclusão

Nesta consulta o esquema relacional conseguiu manter o tempo de execução total (elapsed) igual ao do objeto-relacional mas realizou mais acesso a disco e utilizou mais recurso de CPU.

6.5.15 Teste 15

Objetivo

Detalhar os totais do documento agrupando por tipo item da conta.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID                PRPERF1      T_DOCUMENTO
     INDEX RANGE SCAN                          PRPERF1      IDX_DOC_R2
    TABLE ACCESS BY INDEX ROWID                PRPERF1      T_MEIO_ACESSO
     INDEX RANGE SCAN                          PRPERF1      IDX_MEIO_R3
   TABLE ACCESS BY INDEX ROWID                 PRPERF1      T_ITEM_FATURAMENTO
    AND-EQUAL
     INDEX RANGE SCAN                          PRPERF1      IDX_FK_MEIO_ACESSO
     INDEX RANGE SCAN                          PRPERF1      IDX_FK_DOC_ITEM
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.01	0.11	10	34	0	41
total	3	0.01	0.12	10	34	0	41

Tabela 6.34: Valores obtidos para o esquema relacional no teste 15

Esquema Objeto-Relacional

Plano de Execução

```
SELECT STATEMENT, GOAL = RULE
 SORT GROUP BY
  NESTED LOOPS
   NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_MEIO_ACESSO
          INDEX RANGE SCAN                     PRPERF1      IDX_MEIO5
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OT_DOCUMENTO
          INDEX UNIQUE SCAN                    PRPERF1      SYS_C0019524
    TABLE ACCESS BY INDEX ROWID                PRPERF1      OTN_ITEM_FATURAMENTO
          INDEX RANGE SCAN                     PRPERF1      IDX_ITEM_FK
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.02	0.03	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.00	0.04	3	27	0	41
total	3	0.02	0.07	3	27	0	41

Tabela 6.35: Valores obtidos para o esquema objeto-relacional no teste 15

Conclusão

A estrutura complexa de documento criada no esquema objeto realmente supera o esquema relacional na recuperação dos dados.

Nesta consulta o esquema objeto volta a apresentar melhor tempo de CPU e elapsed, mantendo ainda menos acesso ao disco.

6.5.16 Teste 16

Objetivo

Alteração no valor líquido de um item da conta.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema Relacional

Plano de Execução

```
UPDATE STATEMENT, GOAL = RULE
 UPDATE    PRPERF1        T_ITEM_FATURAMENTO
  INDEX RANGE SCAN       PRPERF1        IDX_FK_DOC_ITEM
```

Análise

Call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.02	0.03	0	0	0	0
Execute	1	0.00	0.04	3	3	7	5
Fetch	0	0.00	0.00	0	0	0	0
total	2	0.02	0.07	3	3	7	5

Tabela 6.36: Valores obtidos para o esquema relacional no teste 16

Esquema Objeto-Relacional

Plano de Execução

```
UPDATE STATEMENT, GOAL = RULE
 UPDATE    PRPERF1       OTN_ITEM_FATURAMENTO
  INDEX RANGE SCAN                    PRPERF1       IDX_ITEM
   TABLE ACCESS BY INDEX ROWID        PRPERF1       OT_DOCUMENTO
         INDEX UNIQUE SCAN            PRPERF1       SYS_C0019522
```

Análise

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.03	0.01	0	0	0	0
Execute	1	0.00	0.04	4	11	5	5
Fetch	0	0.00	0.00	0	0	0	0
total	2	0.03	0.05	4	11	5	5

Tabela 6.37: Valores obtidos para o esquema objeto-relacional no teste 16

Conclusão

Embora a alteração de dados não seja o foco do nosso protótipo, realizamos vários testes e concluímos que a estrutura complexa de documento criada no esquema objeto realmente supera o esquema relacional em desempenho, mas exige maior processamento e acesso a disco quando submetida a uma operação de UPDATE a um item específico dentro da coleção aninhada.

Caso o UPDATE seja direcionado para todas os itens de determinado Documento, a performance tende a ser muito superior, pois o esquema objeto retorna de uma só vez todos os itens do documento em uma única linha, facilitando assim a operação.

6.5.17 Teste 17

Objetivo

Carga de dados e atualização de índices.

Parâmetros alterados

Foram mantidos todos os parâmetros originais

Resultados Obtidos

Esquema	Tempo de Execução (hh:mm:ss)
Relacional	00:14:23
Objeto-Relacional	00:19:15

Tabela 6.38: Valores obtidos no teste 17

Conclusão

Obtivemos uma diferença aproximada de 26% no tempo de carga a favor do esquema relacional. Isto se deve ao custo envolvido na operação de organização dos dados feita durante a carga do esquema objeto-relacional, em que são realizadas varias consultas a tabelas do sistema. Em contrapartida, os resultados referentes à atualização de índices foram muito favoráveis como conseqüência da redução das estruturas de índices.

6.6 Resumo

Este capítulo apresentou toda a estrutura criada para suportar os testes de desempenho e demonstrar os resultados das análises comparativas realizadas entre o esquema relacional e o objeto-relacional.

Podemos observar na tabela abaixo uma sumarização dos resultados dos testes realizados para análise.

Caso	CPU			Elapsed			Disk			Query			Current		
Teste	Rel	Obj	%	Rel	Obj	%	Rel	Obj	%	Rel	Obj	%	Rel	Obj	%
1	0,05	0,04	80%	0,06	0,06	100%	2	2	100%	2	2	100%	0	0	100%
2	0,16	0,03	19%	0,17	0,03	18%	2	2	100%	11080	1770	16%	0	0	100%
3	0,05	0,03	60%	0,05	0,05	100%	0	0	100%	60	54	90%	0	0	100%
4	0,01	0,01	100%	0,02	0,01	50%	0	0	100%	252	256	102%	0	0	100%
5	0,52	0,09	17%	0,71	0,13	18%	2	1	50%	15920	345	2%	0	0	100%
6	0,08	0,04	50%	0,14	0,05	36%	6	2	33%	500	20	4%	0	0	100%
7	1,27	0,01	1%	1,41	0,02	1%	1	2	200%	76638	38	0%	7	0	0%
8	0,04	0,01	25%	0,10	0,03	30%	6	5	83%	250	10	4%	7	0	0%
9	1,71	0,01	1%	1,73	0,05	3%	6	2	33%	101973	56	0%	7	0	0%
10	0,04	0,01	25%	0,12	0,05	42%	11	4	36%	117	32	27%	0	0	100%
11	0,02	0,02	100%	0,11	0,04	36%	8	3	38%	28	28	100%	0	0	100%
12	0,01	0,03	300%	0,08	0,06	75%	9	4	44%	206	32	16%	0	0	100%
13	0,03	0,01	33%	0,12	0,07	58%	2	1	50%	27	42	156%	0	0	100%
14	0,04	0,01	25%	0,04	0,04	100%	2	1	50%	20	14	70%	0	0	100%
15	0,01	0,02	200%	0,12	0,07	58%	10	3	30%	34	27	79%	0	0	100%
16	0,02	0,03	150%	0,07	0,05	71%	3	4	133%	3	11	367%	7	5	71%
Média	0,25	0,03	10%	0,32	0,05	16%	4,4	2,3	51%	12944	171	1%	1,8	0,3	18%

Tabela 6.39: Sumarização dos testes

A tabela acima demonstra de forma detalhada todos os resultados obtidos em cada teste executado. Podemos observar isoladamente para cada teste os valores das métricas coletadas para os dois esquemas, a porcentagem de recurso que o esquema objeto-relacional ocupa em relação ao relacional e uma média final dos valores das métricas e das porcentagens.

O gráfico abaixo foi gerado a partir de médias tiradas dos valores coletados nos testes realizados neste capítulo. Através dele, podemos visualizar de forma sumarizada o resultado do comparativo entre os dois esquemas.

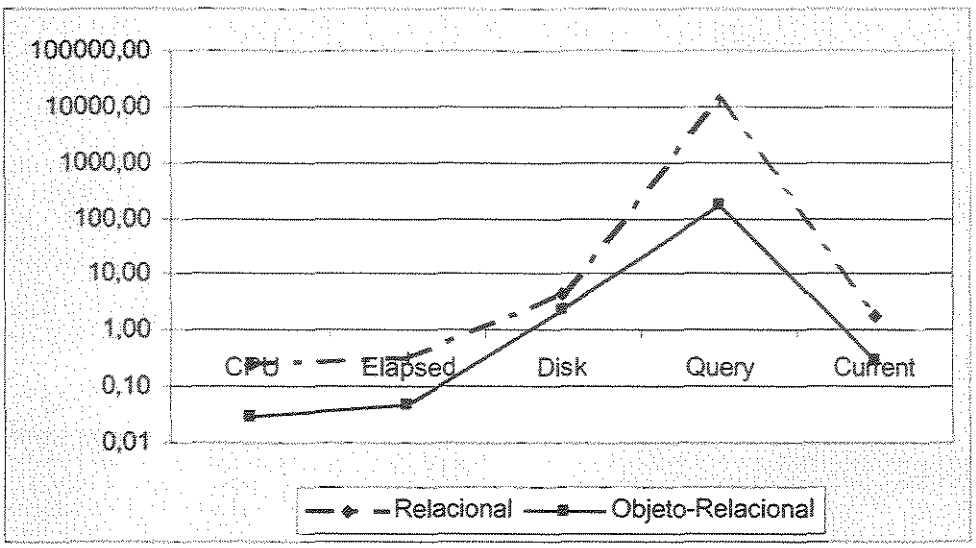

Figura 6.1: Representação gráfica dos resultados dos testes

No decorrer do trabalho identificamos algumas vantagens e desvantagens do modelo Objeto-Relacional sobre o Relacional.

Listaremos abaixo algumas das vantagens identificadas:

- Maior desempenho na recuperação e atualização dos dados;
- Redução do número de acesso a disco quando aplicadas operações de recuperação de dados, pois os dados são alocados por uma coluna da tabela aninhada chamada NESTED_TABLE_ID. Sendo assim, a recuperação dos dados é feita com menos I/O físico, pois os dados encontram-se com pouca fragmentação.
- Maior organização no armazenamento dos dados.
- Diminuição do espaço ocupado em disco através da eliminação de repetições de chaves identificadoras. Com isto as estruturas de índices são beneficiadas com uma redução significativa, o que significa que durante a realização de consultas uma maior parte da estrutura de índices pode ser passada para a memória, diminuindo assim os custos de acesso a disco.

Em contrapartida, as seguintes desvantagens foram encontradas:

- Tempo de carga excessivamente alto;
- Maior complexidade na criação do esquema e de comandos SQL para pessoas não familiarizadas com a Orientação a Objetos.

Capítulo 7

Conclusão

Alguns autores [CD96, SARA98] acreditam que o Modelo Objeto-Relacional será o sucessor do paradigma Relacional ao longo do tempo. Isto se deve principalmente ao fato do modelo Objeto-Relacional preservar as consagradas características relacionais e possibilitar a utilização de recursos providos pela Orientação a Objetos dentro do banco de dados. Estes recursos enriquecem a modelagem de dados e a arquitetura de armazenamento a ponto de atender com mais naturalidade as demandas das novas tecnologias do mercado. Mesmo assim, a penetração desta nova tecnologia no mercado está sendo lenta, pois o modelo Relacional possui características consolidadas como robustez, confiabilidade, performance, capacidade de armazenamento, entre outras que reforçam a resistência às mudanças, mas com o crescimento da tecnologia Orientada a Objetos na área de desenvolvimento de software, será inevitável que os bancos de dados acompanhem esta tecnologia.

Como conclusão dos testes e análises realizadas entre os dois esquemas apresentados no trabalho, pode-se verificar que o emprego do modelo Objeto-Relacional ao protótipo de visualização de contas estudado trouxe consideráveis ganhos de desempenho nas operações de recuperação e atualização de dados, mas apresentou-se ainda imaturo nas operações de inserção, onde o esquema relacional demonstrou-se superior. Na área de modelagem, o modelo Objeto-Relacional esbanja recursos e permite a construção de modelos mais ricos e representativos.

O trabalho teve o propósito de contribuir na demonstração do emprego desta nova tecnologia, e demonstrar que os recursos de modelagem e desempenho deste novo modelo são comparáveis ou até superiores ao consagrado modelo Relacional. Podemos dizer isto, pois o uso da UML na área de modelagem de dados enriquece a representação gráfica de problemas com maior naturalidade e padronização. Quanto ao desempenho, realizamos inúmeros testes focando operações de recuperação, alteração e inserção de dados nesta mesma ordem, pois foi à ordem de prioridade imposta pelas operações do protótipo EBV. Conseguimos então, através dos resultados destes testes, constatar que o desempenho do SGBD Oracle na tecnologia Objeto-Relacional se fez superior nos quesitos de consultas e alterações de dados, perdendo em pouca escala nas

operações de inserção. Com isto, podemos seguramente reforçar o uso desta tecnologia no que diz respeito a modelagem de dados e desempenho.

Como pesquisas futuras, gostaria de propor uma análise das potencialidades e de desempenho dos recursos do modelo Objeto-Relacional como meio de estender a linguagem PL/SQL em Oracle. O tipo de objeto é uma excelente maneira de estender a linguagem com novas funcionalidades, do mesmo modo que uma estrutura de classe faz isso em C++ ou Java. Além disso, poderiam ser feitos estudos de estratégias para migração de bases de dados relacionais para esta nova tecnologia o que reforçaria ainda mais o emprego dessa nova tecnologia no mercado.

Referências

[ARRU00] Arruda, Rodrigo M. S. Analise de Mapeamento de Classes em Tabelas em Banco de Dados Objeto – Relacional, UFPE, 2000 (Dissertação de Mestrado).

[BIND00] Binder, Robert "*Testing Objec t Oriented Systems: Models, Patterns and Tools*". Addison Wesley 2000.

[BM93] Bertino, E. and Martino L., Object Oriented Database Systems: Concepts and Architecture, Addison-Wesley (1993).

[BP98] Blaha, M. and Premerlani, W. "*Object-Oriented Modeling and Design for Database Applications* ". Prentice Hall (1998).

[CD96] Carey, M. J. and DeWitt, D. J.. "Of Objects and Databases: A decade of Turmoil". Proceedings of the 22th International Conference on Very Large Databases. (September 1996)

[DATE91] DATE, C.J, "*Introdução a sistemas de Banco de Dados*". 4ª edição. Rio de Janeiro. Ed. Campus Ltda, 1991. 674p. p 99-105.

[DORN] DORNELES, C. Banco de Dados Objeto-Relacional. Porto Alegre: UFRGS – Pós-Graduação em Ciência da Computação. Disponível www. inf.ufrgs.br/~carina

[EM99] Eisenberg, A. and Melton, J.. "SQL : 1999, formerly known as SQL3." SIGMODRecord 28(1): 131-138 (1999).

[EN94] Elmasri, R. and S. B. Navathe. *Fundamentals of Database System*, The Benjamin/Cummings Publishing Company, Inc. (1994).

[FERN8I] Fernandes, Lúcia, "*ORACLE DEVELOPER – ORACLE 8/8i*", Ed. AXCEL BOOKS.

[FK00] Fowler, Martin and Kendall, S., "UML Essencial". 2ª Edição. Porto Alegre. Ed. Bookman.

[GZS00] GUERRA, D. C. and ZAMBAN, L,B. and SANTOS, M. S. "*Banco de Dados Objeto- Relacionais*". Porto Alegre: PUCRS – Disponível: www.inf.pucrs.br/~arruda/artigos/gpbd_991/grupo04/index.htm

[INFO98] Informix Corporation. "*OnLine Dynamic Server Databases.*" Disponivel: www.informix.com (2002).

[KYTE8I] Kyte, Thomas, "*Dominando Oracle – ORACLE 7.3/8.x*", Ed.CIÊNCIA MODERNA.

[MCFA99] McFadden, Fred R. "Modern Database Management", *5ª Ed. Addison-Wesley Educational Publishers Inc.*

[ORA9I] Oracle Corporation. *Manual do SGBD Objeto-Relacional Oracle 9i.*

[RH96] Ramanathan, S. and Hodges, J. "Reverse Engineering Relational Schemas to Object-Oriented Schemas". Technical Report. Departament of Computer Science. Mississipi State Science (July 1996).

[RH97] Ramanathan, S. and Hodges, J.. "Extraction of Object-Oriented Structures from Existing Relational Databases." SIGMOD Record 26(1): 59-64 (March 1997).

[RUMB91] Rumbaugh, J. *Object Oriented Modeling and Design.* Prentice Hall, Inc, 1991.

[SALG99] SALGADO, Ana Carolina. "*In Simpósio Brasileiro de Banco de Dados*", SGBD Objeto-Relacional. Florianópolis. 1999

[SARA98] Saracco, C. M. Universal Database Management. A Guide to Object/Relational Technology. Morgan Kaufmann Publishers. San Francisco, California (1998).

[SILV98] SILVA, A. P. M. "*Migrando Banco de Dados Relacionais para Tecnologia Objeto Relacional*". UFMG: Pós-Graduação em Informática. Disponível: www.dcc.ufmg.br/pos/html/spg98/anais/alecia/alecia.html

[SK98] Silberschatz, A. and Korth, H. F., et al. Database System Concepts (1998).

[SK99] Silberschatz, A. and Korth, Banco de Dados Relacionais-Objeto. *In: Sistemas de Bancos de Dados*. 3. ed. São Paulo: Makron Books, 1999. p.273-289

[SM96] Stonebraker, M. and Moore, D.. Object-Relational DBMS. The Next Great Wave, Morgan Kaufmann Publishers (1996).

[TO97] Traduzindo Objetos em Bancos de Dados Relacional. *Developers Magazine*. ano 2, n 14, outubro, 1997, p 12-13.

[WILH99] Wilhelms, E. , Administração de Sistemas Gerenciadores de Banco de Dados Objeto Relacional em Ambiente Distribuído, UCB, 1999, (Dissertação de Mestrado).

Anexo A

Comandos de criação da base de dados Relacional

Nesta seção, apresentaremos os comandos de criação da base de dados, baseados no modelo apresentado na seção 4.2.

```
CREATE TABLE "SESSAO"
(
"CD"                                NUMBER(12)              NOT NULL,
"DT_HORA_INICIO"                    DATE                    NOT NULL,
"ENDERECO_REMOTO"                   VARCHAR2(100)           NOT NULL,
"FK_USUARIO_CD"                     NUMBER(12)              NOT NULL,
"MEIO_ACESSO"                       VARCHAR2(290)           NOT NULL,
CONSTRAINT "PK_SESSAO"              PRIMARY KEY ("CD")
)
/

CREATE TABLE "USUARIO"
(
"NM"                                VARCHAR2(64),
"CD_FATURAMENTO"                    VARCHAR2(45)            NOT NULL,
"SENHA"                             VARCHAR2(30)            NOT NULL,
"EMAIL"                             VARCHAR2(100),
"CONFIRMACAO"                       VARCHAR2(50),
"DT_BLOQUEIO"                       DATE,
"DT_CRIACAO"                        DATE NOT NULL,
"DT_ALTERACAO_SENHA"                DATE,
"DE_LOGRADOURO"                     VARCHAR2(100),
"NUMERO"                            VARCHAR2(50),
"PONTO_REFERENCIA"                  VARCHAR2(200),
"CD"                                NUMBER(12)              NOT NULL,
"MEIO_ACESSO"                       VARCHAR2(290)           NOT NULL,
"CPFCGC"                            VARCHAR2(15),
"TP_LOGRADOURO"                     VARCHAR2(15),
 CONSTRAINT "PK_USUARIO"            PRIMARY KEY ("CD")
)
/


CREATE TABLE "DOCUMENTO"
(
"NR_SERIE_NOTA_FISCAL"              VARCHAR2(15),
"NR_NOTA_FISCAL"                    NUMBER(15,0),
"CD_CONTROLE_NOTA_FISCAL"           VARCHAR2(15),
"DT_PERIODO_REFERENCIA"             DATE,
"DT_INICIO_REFERENCIA"              DATE                    NOT NULL,
```

```
"DT_FIM_REFERENCIA"             DATE                    NOT NULL,
"DT_VENCIMENTO"                 DATE,
"DT_CICLO"                      DATE,
"VL_CORRECAO_PAGAMENTO"         NUMBER(15,2),
"VL_CONTA_ANTERIOR"             NUMBER(15,2),
"VL_CONTA_ANTERIOR_PAGO"        NUMBER(15,2),
"VL_AJUSTES"                    NUMBER(15,2),
"CD"                            NUMBER(12,0)            NOT NULL,
"VL_CONSUMO"                    NUMBER(15,2),
"TP_REMESSA"                    NUMBER(1)               NOT NULL,
"DT_GERACAO_ARQUIVO"            DATE NOT NULL,
"NUM_FAT_AGREGADOR"             NUMBER(11,0)            NOT NULL,
 CONSTRAINT "PK_DOCUMENTO"      PRIMARY KEY ("CD")
)
/

CREATE TABLE "MEIO_ACESSO"
(
"NR_MEIO_ACESSO_AGREGADO"       VARCHAR2(290)           NOT NULL,
"NUM_FAT"                       NUMBER(11,0)            NOT NULL,
"TP_MEIO_ACESSO"                VARCHAR2(5),
"NM_CLIENTE"                    VARCHAR2(64),
"CPF_CGC"                       NUMBER(14,0),
"TP_LOGRADOURO"                 VARCHAR2(6),
"TITULO_LOGRADOURO"             VARCHAR2(7),
"NUMERO"                        NUMBER(5),
"NM_LOGRADOURO"                 VARCHAR2(64),
"PONTO_REFERENCIA"              VARCHAR2(500),
"NM_LOCALIDADE_NACIONAL"        VARCHAR2(40),
"SIGLA_UF"                      VARCHAR2(2),
"CEP"                           NUMBER(5),
"COMPLEMENTO_CEP"               NUMBER(4),
"NRC"                           NUMBER(14,0),
"NR_CAIXA_POSTAL"               NUMBER(5),
"COMPLEMENTO_ENDERECO"          VARCHAR2(80),
"FK_DOCUMENTO_CD"               NUMBER(12,0)            NOT NULL,
"CD"                            NUMBER(12)              NOT NULL,
 CONSTRAINT "PK_MEIO_ACESSO"    PRIMARY KEY ("CD")
)
/

CREATE TABLE "ITEM_DA_CONTA"
(
"ITEM_FATURAMENTO"              NUMBER(12)              NOT NULL,
"DOMINIO"                       NUMBER(10),
"DE_SERVICO"                    VARCHAR2(60),
"DT_ITEM"                       DATE,
"HR_ITEM"                       DATE,
"NR_TERMINAL_DESTINO"           VARCHAR2(19),
"NR_TERMINAL_ORIGEM"            VARCHAR2(19),
"DURACAO_TARIFADA"              NUMBER(6,2),
"LOCALIDADE_ORIGEM"             VARCHAR2(18),
"LOCALIDADE_DESTINO"            VARCHAR2(18),
"ESTADO_ORIGEM"                 VARCHAR2(1000),
"ESTADO_DESTINO"                VARCHAR2(2),
"GRUPO_HORARIO"                 VARCHAR2(3),
"CALLING_CARD"                  VARCHAR2(40),
```

```
"UNIDADE_MEDIDA"                    VARCHAR2(2),
"QTDE_SERVICO"                      NUMBER(10),
"VL_BRUTO"                          NUMBER(15,2),
"VL_LIQUIDO"                        NUMBER(15,2),
"CD" NUMBER(12)                     NOT NULL,
"MNEMONICO_PLANO"                   CHAR(1),
"ID_AJUSTE"                         CHAR(1),
"DE_BLOCO_SERVICO"                  VARCHAR2(40),
"CD_PRIORIDADE_BLOCO"               NUMBER(2),
"FK_BLOCO_SERVICO_CD"               NUMBER(12)              NOT NULL,
"FK_MEIO_ACESSO_CD"                 NUMBER(12)              NOT NULL,
"FK_DOCUMENTO_CD"                   NUMBER(12)              NOT NULL,
  CONSTRAINT "PK_ITEM_FATURAMENTO"  PRIMARY KEY ("CD")
)
/

CREATE TABLE "GRUPO_DE_ITENS"
(
"DE_BLOCO_SERVICO"                  VARCHAR2(40),
"VL_TOTAL_BLOCO_SERVICO"            NUMBER(15,2),
"CD_PRIORIDADE_BLOCO"               NUMBER(2),
"CD"                                NUMBER(12)              NOT NULL,
"FK_MEIO_ACESSO_CD"                 NUMBER(12)              NOT NULL,
  CONSTRAINT "PK_BLOCO_SERVICO"     PRIMARY KEY ("CD")
)
/

CREATE TABLE "VISAO"
(
"CD"                                NUMBER(12,0)            NOT NULL,
"NM"                                VARCHAR2(50),
"ROTULO"                            VARCHAR2(50),
"ID_DISPONIVEL"                     CHAR(1),
  CONSTRAINT "PK_VISAO"             PRIMARY KEY ("CD")
)
/

CREATE TABLE "VISAO_IMPL"
(
"CD"                                NUMBER(12,0)            NOT NULL,
"DT_INICIO"                         DATE,
"DT_FIM"                            DATE,
"NM_CLASSE_COMANDO"                 VARCHAR2(250),
"NM_COMPONENTE_WEB"                 VARCHAR2(250),
"FK_VISAO_CD"                       NUMBER(12,0)            NOT NULL,
  CONSTRAINT "PK_VISAO_IMPL"        PRIMARY KEY ("CD")
)
/

ALTER TABLE "SESSAO" ADD CONSTRAINT "USR_SSN" FOREIGN KEY (
"FK_USUARIO_CD")
REFERENCES "USUARIO" ("CD") ON DELETE CASCADE
/

ALTER TABLE "MEIO_ACESSO" ADD CONSTRAINT "DCM_MEIO" FOREIGN KEY (
"FK_DOCUMENTO_CD")
REFERENCES "DOCUMENTO" ("CD") ON DELETE CASCADE
```

```
/

ALTER TABLE "ITEM_DA_CONTA" ADD CONSTRAINT "BLOCO_ITEM" FOREIGN KEY
("FK_BLOCO_SERVICO_CD")
REFERENCES "BLOCO_SERVICO" ("CD") ON DELETE CASCADE
/

ALTER TABLE "ITEM_DA_CONTA" ADD CONSTRAINT "MEIO_ITEM" FOREIGN KEY
("FK_MEIO_ACESSO_CD")
REFERENCES "MEIO_ACESSO" ("CD") ON DELETE CASCADE
/

ALTER TABLE "ITEM_DA_CONTA" ADD CONSTRAINT "DOCUMENTO_ITEM" FOREIGN KEY
("FK_DOCUMENTO_CD")
REFERENCES "DOCUMENTO" ("CD") ON DELETE CASCADE
/

ALTER TABLE "GRUPO_DE_ITENS" ADD CONSTRAINT "MEIO_BLOCO" FOREIGN KEY
("FK_MEIO_ACESSO_CD")
REFERENCES "MEIO_ACESSO" ("CD") ON DELETE CASCADE
/

ALTER TABLE "VISAO_IMPL" ADD CONSTRAINT "VSO_VIMP" FOREIGN KEY (
"FK_VISAO_CD")
REFERENCES "VISAO" ("CD")
/
```

Anexo B

Comandos de criação da base de dados Objeto-Relacional

Tipos de Objetos

```
create or replace type O_VISAO_IMPL as object(
 CD                    NUMBER(12),
 DT_INICIO             DATE,
 DT_FIM                DATE,
 NM_CLASSE_COMANDO     VARCHAR2(250),
 NM_COMPONENTE_WEB     VARCHAR2(250)
)
/

create or replace type ON_VISAO_IMPL as table of O_VISAO_IMPL
/

create or replace type O_VISAO as object(
 CD                NUMBER(12) ,
 NM                VARCHAR2(50),
 ROTULO            VARCHAR2(50),
 ID_DISPONIVEL     CHAR(1),
 V_IMPL            ON_VISAO_IMPL
)
/

create or replace type O_USUARIO as object(
 NM                        VARCHAR2(64),
 CD_FATURAMENTO            VARCHAR2(45),
 SENHA                     VARCHAR2(30),
 EMAIL                     VARCHAR2(100),
 CONFIRMACAO               VARCHAR2(50),
 DT_BLOQUEIO               DATE,
 DT_CRIACAO                DATE,
 DT_ALTERACAO_SENHA        DATE,
 DE_LOGRADOURO             VARCHAR2(100),
 NUMERO                    NUMBER(10),
 ESTADO                    VARCHAR2(20),
 CEP                       VARCHAR2(10),
 CAIXA_POSTAL              NUMBER(10),
 CIDADE                    VARCHAR2(50),
 COMPLEMENTO               VARCHAR2(50),
 PONTO_REFERENCIA          VARCHAR2(200),
 CD                        NUMBER(12),
 MEIO_ACESSO               VARCHAR2(290),
 CPFCGC                    VARCHAR2(15),
 TP_LOGRADOURO             VARCHAR2(15)
```

```
)
/

create or replace type O_SESSAO as object(
 CD                  NUMBER(12),
 DT_HORA_INICIO      DATE,
 ENDERECO_REMOTO     VARCHAR2(100),
 MEIO_ACESSO         VARCHAR2(290),
 RF_USUARIO          REF O_USUARIO
)
/

create or replace type O_MEIO_ACESSO as object(
 NR_MEIO_ACESSO_AGREGADO  VARCHAR2(290),
 NUM_FAT                  NUMBER(11),
 TP_MEIO_ACESSO           VARCHAR2(5),
 NM_CLIENTE               VARCHAR2(64),
 CPF_CGC                  NUMBER(14),
 TP_LOGRADOURO            VARCHAR2(6),
 TITULO_LOGRADOURO        VARCHAR2(7),
 NUMERO                   NUMBER(5),
 NM_LOGRADOURO            VARCHAR2(64),
 PONTO_REFERENCIA         VARCHAR2(500),
 NM_LOCALIDADE_NACIONAL   VARCHAR2(40),
 SIGLA_UF                 VARCHAR2(2),
 CEP                      NUMBER(5),
 COMPLEMENTO_CEP          NUMBER(4),
 NRC                      NUMBER(14),
 NR_CAIXA_POSTAL          NUMBER(5),
 COMPLEMENTO_ENDERECO     VARCHAR2(80),
 CD                       NUMBER(12)
 )
/

create or replace type O_GRUPO_DE_ITENS as object(
 DE_BLOCO_SERVICO          VARCHAR2(40),
 VL_TOTAL_BLOCO_SERVICO    NUMBER(15,2),
 CD_PRIORIDADE_BLOCO       NUMBER(2),
 CD                        NUMBER(12),
 RF_MEIO_ACESSO            REF O_MEIO_ACESSO
 )
/

create or replace type O_ITEM_DA_CONTA as object(
 ITEM_FATURAMENTO     NUMBER(12),
 DOMINIO              NUMBER(10),
 DE_SERVICO           VARCHAR2(60),
 NR_TERMINAL_DESTINO  VARCHAR2(19),
 NR_TERMINAL_ORIGEM   VARCHAR2(19),
 DURACAO_TARIFADA     NUMBER(6,2),
 LOCALIDADE_ORIGEM    VARCHAR2(18),
 LOCALIDADE_DESTINO   VARCHAR2(18),
 ESTADO_ORIGEM        VARCHAR2(1000),
 ESTADO_DESTINO       VARCHAR2(2),
 GRUPO_HORARIO        VARCHAR2(3),
 CALLING_CARD         VARCHAR2(40),
 UNIDADE_MEDIDA       VARCHAR2(2),
```

```
 QTDE_SERVICO          NUMBER(10),
 VL_BRUTO              NUMBER(15,2),
 VL_LIQUIDO            NUMBER(15,2),
 CD                    NUMBER(12),
 MNEMONICO_PLANO       CHAR(1),
 ID_AJUSTE             CHAR(1),
 DE_BLOCO_SERVICO      VARCHAR2(40),
 DT_ITEM               DATE,
 HR_ITEM               DATE,
 CD_PRIORIDADE_BLOCO   NUMBER(2),
 FK_MEIO_ACESSO_CD     NUMBER(12),
 RF_BLOCO              REF O_GRUPO_DE_ITENS
)
/

create or replace type ON_ITEM_DA_CONTA as table of O_ITEM_DA_CONTA
/

create or replace type ON_MEIO_ACESSO as table of O_MEIO_ACESSO
/

create or replace type O_DOCUMENTO as object(
 NR_SERIE_NOTA_FISCAL          VARCHAR2(15),
 NR_NOTA_FISCAL                NUMBER(15),
 CD_CONTROLE_NOTA_FISCAL       VARCHAR2(15),
 DT_PERIODO_REFERENCIA         DATE,
 DT_INICIO_REFERENCIA          DATE,
 DT_FIM_REFERENCIA             DATE,
 DT_VENCIMENTO                 DATE,
 DT_CICLO                      DATE,
 VL_CORRECAO_PAGAMENTO         NUMBER(15,2),
 VL_CONTA_ANTERIOR             NUMBER(15,2),
 VL_CONTA_ANTERIOR_PAGO        NUMBER(15,2),
 VL_AJUSTES                    NUMBER(15,2),
 CD                            NUMBER(12),
 VL_CONSUMO                    NUMBER(15,2),
 TP_REMESSA                    NUMBER(1),
 DT_GERACAO_ARQUIVO            DATE,
 NUM_FAT_AGREGADOR             NUMBER(11),
 SALDO_TOTAL_ANTERIOR          NUMBER(15,2),
 MEIO                          ON_MEIO_ACESSO,
 ITEM                          ON_ITEM_DA_CONTA
)
/
```

Tabelas Objeto

```
create table OT_VISAO of O_VISAO
(CD                          primary key)
object identifier is primary key
nested table V_IMPL store as OTN_VISAO_IMPL;

create index IDX_IMPL    on OTN_VISAO_IMPL(nested_table_id, CD);
```

```
create table OT_USUARIO of O_USUARIO
(CD                        primary key,
 CD_FATURAMENTO   not null,
 SENHA           not null,
 DT_CRIACAO         not null,
 MEIO_ACESSO      not null);
create index IDX_USER1 on OT_USUARIO(MEIO_ACESSO);


create table OT_SESSAO of O_SESSAO
(CD primary Key,
 DT_HORA_INICIO  not null,
 ENDERECO_REMOTO not null,
 MEIO_ACESSO     not null,
 RF_USUARIO      WITH ROWID REFERENCES OT_USUARIO);


create table OT_GRUPO_DE_ITENS of O_BLOCO_DE_ITENS
(CD         primary key);


create table OT_DOCUMENTO of O_DOCUMENTO
(CD         primary key,
 DT_INICIO_REFERENCIA    not null,
 DT_FIM_REFERENCIA       not null,
 TP_REMESSA              not null,
 DT_GERACAO_ARQUIVO      not null,
 NUM_FAT_AGREGADOR       not null)
 object identifier is primary key
 nested table MEIO store as OTN_MEIO_ACESSO
 nested table ITEM store as OTN_ITEM_DA_CONTA;
```

Index

```
create index IDX_DOC     on OT_DOCUMENTO (nr_nota_fiscal);
create index IDX_DOC1    on OT_DOCUMENTO (DT_INICIO_REFERENCIA);
create index IDX_MEIO    on OTN_MEIO_ACESSO(nested_table_id, CD);
create index IDX_MEIO2   on OTN_MEIO_ACESSO(NRC);
create index IDX_MEIO3   on OTN_MEIO_ACESSO(num_fat,nr_meio_acesso_agregado);
create index IDX_MEIO4   on OTN_MEIO_ACESSO(cpf_cgc);
create index IDX_MEIO5   on OTN_MEIO_ACESSO(nr_meio_acesso_agregado);
create index IDX_ITEM    on OTN_ITEM_DA_CONTA(nested_table_id, CD);
create index IDX_ITEM2   on OTN_ITEM_DA_CONTA(dominio);
create index IDX_ITEM_FK on OTN_ITEM_DA_CONTA(FK_MEIO_ACESSO_CD);
```

Anexo C

Comandos da carga de trabalho testada.

1) Recuperar informações de usuários para simples conferência, através do número do meio de acesso (Telefone).

```
-- RELACIONAL
select u.CD,u.MEIO_ACESSO,u.CPFCGC,u.NM,u.EMAIL,u.CONFIRMACAO,
       u.DT_BLOQUEIO Data_Hora_Bloqueio, u.DT_CRIACAO Data_Hora_Criacao,
       u.DT_ALTERACAO_SENHA,
       (u.TP_LOGRADOURO || ' ' || u.DE_LOGRADOURO) AS LOGRADOURO,
       u.NUMERO,u.ESTADO,u.CEP,u.CAIXA_POSTAL,u.CIDADE,
       u.COMPLEMENTO,u.PONTO_REFERENCIA,
       u.CD_FATURAMENTO
from  t_USUARIO u
where u.MEIO_ACESSO= '&meio_acesso';

-- OBJETO
select u.CD,u.MEIO_ACESSO,u.CPFCGC,u.NM,u.EMAIL,u.CONFIRMACAO,
       u.DT_BLOQUEIO Data_Hora_Bloqueio, u.DT_CRIACAO Data_Hora_Criacao,
       u.DT_ALTERACAO_SENHA,
       (u.TP_LOGRADOURO || ' ' || u.DE_LOGRADOURO) AS LOGRADOURO,
       u.NUMERO,u.ESTADO,u.CEP,u.CAIXA_POSTAL,u.CIDADE,u.COMPLEMENTO,
       u.PONTO_REFERENCIA, u.CD_FATURAMENTO
from  OT_USUARIO u
where u.MEIO_ACESSO= '&meio_acesso';
```

2) Recuperar informações de notas fiscais de determinado cliente, utilizando o CPF como chave de busca.

```
-- RELACIONAL
Select distinct m.nrc, m.nr_meio_acesso_agregado AS Agregador,
       d.nr_nota_fiscal,(d.vl_consumo+d.vl_correcao_pagamento+
       d.vl_ajustes+d.saldo_total_anterior) VlTotal,
       d.saldo_total_anterior as Balance,m.num_fat,
       m.fk_documento_cd,  d.dt_inicio_referencia
from   t_meio_acesso m, t_documento d
where m.fk_documento_cd = d.cd
and   m.num_fat = d.num_fat_agregador
and   m.fk_documento_cd in
     (select distinct d.cd
      from t_documento d, t_meio_acesso m
      where d.cd = m.fk_documento_cd
      and   m.cpf_cgc= '&CPF_CGC'
order by d.dt_inicio_referencia DESC;
```

```
-- OBJETO
Select distinct m.nrc, m.nr_meio_acesso_agregado AS Agregador,
       d.nr_nota_fiscal,(d.vl_consumo+d.vl_correcao_pagamento+
       d.vl_ajustes+d.saldo_total_anterior) VlTotal,
       d.saldo_total_anterior as Balance, m.num_fat,
       d.cd, d.dt_inicio_referencia
from  ot_documento d, table(d.meio) m
where m.num_fat = d.num_fat_agregador
and   d.cd in
     (select distinct d.cd
      from ot_documento d, table(d.meio) m
      where m.cpf_cgc= '&CPF_CGC')
order by d.dt_inicio_referencia DESC;
```

3) Recupera para consulta informações da nota fiscal de um meio de acesso específico.

```
-- RELACIONAL
select distinct m.nrc, m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,
                (d.vl_consumo +
d.vl_correcao_pagamento+d.vl_ajustes+d.saldo_total_anterior) VlTotal,
                d.saldo_total_anterior as Balance,
                m.num_fat, m.fk_documento_cd,d.dt_inicio_referencia
from  t_meio_acesso m, t_documento d
where m.fk_documento_cd = d.cd
and   m.num_fat = d.num_fat_agregador
and   m.fk_documento_cd in
      (select d.cd
      from t_documento d, t_meio_acesso m
      where
      d.cd = m.fk_documento_cd
      and m.nr_meio_acesso_agregado = '&meio_acesso')
order by d.dt_inicio_referencia DESC;

-- OBJETO
select distinct m.nrc, m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,
                (d.vl_consumo +
d.vl_correcao_pagamento+d.vl_ajustes+d.saldo_total_anterior) VlTotal,
                d.saldo_total_anterior as Balance,
                m.num_fat, d.cd,d.dt_inicio_referencia
from  ot_documento d, table(d.meio) m
where m.num_fat = d.num_fat_agregador
and d.cd in
      (select d.cd
      from ot_documento d, table(d.meio) m
      where m.nr_meio_acesso_agregado = '&meio_acesso')
order by d.dt_inicio_referencia DESC;
```

4) Selecionando informações do documento mais recente para determinado cliente, utilizando o CPF/CGC como chave de busca.

```
-- RELACIONAL
select M.NR_MEIO_ACESSO_AGREGADO,(M.TP_LOGRADOURO || ' ' ||
       M.NM_LOGRADOURO || ',' || M.NUMERO || ' ' ||
       M.COMPLEMENTO_ENDERECO) as LogMeio,
       M.PONTO_REFERENCIA PtoRefMeio,(M.NM_LOCALIDADE_NACIONAL ||
       '-' || M.SIGLA_UF || ' ' || M.CEP || '-' ||
       M.COMPLEMENTO_CEP) as LOCMeio,M.NR_CAIXA_POSTAL
from  t_MEIO_ACESSO M, t_DOCUMENTO D
where M.FK_DOCUMENTO_CD=D.CD
and   M.CPF_CGC = '&CPF_CGC'
and   D.DT_INICIO_REFERENCIA =
      (SELECT MAX(Do.DT_INICIO_REFERENCIA)
      FROM t_DOCUMENTO do, t_meio_acesso me
      WHERE do.CD = me.fk_documento_cd
      and   me.cpf_cgc = m.cpf_cgc);

-- OBJETO
select M.NR_MEIO_ACESSO_AGREGADO,(M.TP_LOGRADOURO || ' ' ||
       M.NM_LOGRADOURO || ',' || M.NUMERO || ' ' ||
       M.COMPLEMENTO_ENDERECO) as LogMeio,
       M.PONTO_REFERENCIA PtoRefMeio,(M.NM_LOCALIDADE_NACIONAL ||
       '-' || M.SIGLA_UF || ' ' || M.CEP || '-' ||
       M.COMPLEMENTO_CEP) as LOCMeio,M.NR_CAIXA_POSTAL
from ot_documento d, table(d.meio) m
where  M.CPF_CGC = '&CPF_CGC'
and    D.DT_INICIO_REFERENCIA =
       (SELECT MAX(do.DT_INICIO_REFERENCIA)
        from ot_documento do, table(d.meio) me
        WHERE me.cpf_cgc = m.cpf_cgc);
```

5) Seleciona informações do documento mais recente para determinado meio de acesso.

```
-- RELACIONAL
select M.NR_MEIO_ACESSO_AGREGADO,
      (M.TP_LOGRADOURO || ' ' || M.NM_LOGRADOURO || ',' || M.NUMERO || ' '
      || M.COMPLEMENTO_ENDERECO) as LogMeio,
      M.PONTO_REFERENCIA PtoRefMeio,
      (M.NM_LOCALIDADE_NACIONAL || '-' || M.SIGLA_UF || ' ' || M.CEP || '-' ||
      M.COMPLEMENTO_CEP) as LOCMeio,
      M.NR_CAIXA_POSTAL
from  t_MEIO_ACESSO M, t_DOCUMENTO D
where M.FK_DOCUMENTO_CD=D.CD
AND   M.NR_MEIO_ACESSO_AGREGADO = '&meio_acesso'
and   D.DT_INICIO_REFERENCIA =
      (SELECT MAX(Do.DT_INICIO_REFERENCIA)
      FROM t_DOCUMENTO do, t_meio_acesso me
      WHERE do.CD = me.fk_documento_cd
      and   me.nr_meio_acesso_agregado = m.nr_meio_acesso_agregado);
```

```
-- OBJETO
select M.NR_MEIO_ACESSO_AGREGADO,
       (M.TP_LOGRADOURO || ' ' || M.NM_LOGRADOURO || ',' || M.NUMERO || ' ' ||
       M.COMPLEMENTO_ENDERECO) as LogMeio,
       M.PONTO_REFERENCIA PtoRefMeio,
       (M.NM_LOCALIDADE_NACIONAL || '-' || M.SIGLA_UF || ' ' || M.CEP || '-' ||
       M.COMPLEMENTO_CEP) as LOCMeio,
       M.NR_CAIXA_POSTAL
from   ot_documento d, table(d.meio) m
where M.NR_MEIO_ACESSO_AGREGADO = '&meio_acesso'
and    D.DT_INICIO_REFERENCIA =
       (SELECT MAX(do.DT_INICIO_REFERENCIA)
       FROM ot_documento do, table(do.meio) me
       WHERE me.nr_meio_acesso_agregado = m.nr_meio_acesso_agregado);
```

6) Sumarização das chamadas telefônicas de uma nota fiscal agrupadas por tarifa. Dentre estas informações teremos: Totais de chamadas, duração total e valores líquidos e brutos.

```
-- RELACIONAL
select i.grupo_horario as Tarifa,
       count(i.cd) as TotalChamadas,
       Sum(i.duracao_tarifada) duracaoTotal,
       Sum(i.vl_liquido) as TotLiq, Sum(i.vl_bruto) TotBruto
from   t_item_faturamento i,t_meio_acesso m,t_documento d
where  i.fk_meio_acesso_cd = m.cd
and    i.fk_documento_cd = d.cd
and    i.fk_documento_cd = m.fk_documento_cd
and    i.dominio = 11
and   m.fk_documento_cd = d.cd
and   d.nr_nota_fiscal = '&NumNF'
group by i.grupo_horario;

-- OBJETO
select i.grupo_horario as Tarifa,
       count(i.cd) as TotalChamadas,
       Sum(i.duracao_tarifada) duracaoTotal,
       Sum(i.vl_liquido) as TotLiq, Sum(i.vl_bruto) TotBruto
from   ot_documento d, table(d.item) i
where  i.dominio = 11
and    d.nr_nota_fiscal = '&NumNF'
group by i.grupo_horario;
```

7) Obter o número da nota fiscal de cada documento emitido para um meio de acesso.

```
-- RELACIONAL
select distinct d.nr_nota_fiscal,d.dt_inicio_referencia
DtIni,d.dt_fim_referencia DtFim, d.vl_consumo
from  t_meio_acesso m, t_documento d
where m.fk_documento_cd = d.cd
and   m.num_fat = d.num_fat_agregador
and   m.num_fat in
      (select distinct d.num_fat_agregador
      from t_documento d, t_meio_acesso m
      where
```

```
      d.cd = m.fk_documento_cd
      and  m.nr_meio_acesso_agregado = '&meio_acesso');

-- OBJETO
select distinct d.nr_nota_fiscal,d.dt_inicio_referencia
DtIni,d.dt_fim_referencia DtFim, d.vl_consumo
from  ot_documento d, table(d.meio) m
where m.num_fat = d.num_fat_agregador
and    exists
      (select d.num_fat_agregador
      from ot_documento d, table(d.meio) me
      where me.nr_meio_acesso_agregado = '&meio_acesso'
      and m.num_fat = d.num_fat_agregador);
```

8) Recuperar valores líquidos e brutos de todas as chamadas de um meio de acesso em um determinado período.

```
-- RELACIONAL
select d.dt_inicio_referencia, d.dt_fim_referencia,
       sum(i.vl_liquido) TotChamadaLiq,sum(i.vl_bruto) TotChamadaBruto
from t_item_faturamento i,t_meio_acesso m,t_documento d
where i.fk_meio_acesso_cd = m.cd
and   i.fk_documento_cd = d.cd
and   i.fk_documento_cd = m.fk_documento_cd
and   d.nr_nota_fiscal = '&NumNF'
and   i.cd_prioridade_bloco in (10,11,15,19,22,26,28,30,33,37,50,55)
group by d.dt_inicio_referencia, d.dt_fim_referencia;

--OBJETO
select
d.dt_inicio_referencia, d.dt_fim_referencia,
sum(i.vl_liquido) TotChamadaLiq,sum(i.vl_bruto) TotChamadaBruto
from ot_documento d, table(d.item)i
where d.nr_nota_fiscal = '&NumNF'
and   i.cd_prioridade_bloco in (10,11,15,19,22,26,28,30,33,37,50,55)
group by d.dt_inicio_referencia, d.dt_fim_referencia;
```

9) Recuperar informações do agregador, ou seja, o meio de acesso responsável pelas chamadas de cada documento.

```
-- RELACIONAL
select distinct m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,d.vl_consumo,
                m.num_fat, m.nrc, m.fk_documento_cd
from  t_meio_acesso m, t_documento d
where m.fk_documento_cd = d.cd
and   m.num_fat = d.num_fat_agregador
and   m.num_fat in
      (select distinct d.num_fat_agregador
      from t_documento d, t_meio_acesso m
      where
      d.cd = m.fk_documento_cd
      and  m.nr_meio_acesso_agregado = '&meio_acesso');
```

```
-- OBJETO
select distinct m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,d.vl_consumo,
                m.num_fat, m.nrc, d.cd
from  ot_documento d, table(d.meio)m
where m.num_fat = d.num_fat_agregador
and   m.num_fat in
      (select distinct d.num_fat_agregador
      from ot_documento d, table(d.meio)m
      where m.nr_meio_acesso_agregado = '&meio_acesso');
```

10) Selecionar detalhes de todos os itens da conta de um determinado documento de recebimento de um meio de acesso.

```
-- RELACIONAL
Select i.de_bloco_servico, i.dt_item,
       To_char(i.hr_item,'HH24:MI:SS') HrItem,
       i.localidade_destino || '-' || i.estado_destino as LocalidadeDestino,
       i.de_servico, i.nr_terminal_destino,
       i.grupo_horario,
       i.duracao_tarifada duracao, i.vl_liquido as TotLiq,
       i.vl_bruto TotBruto
from t_item_faturamento i,t_meio_acesso m, t_documento d, t_bloco_servico b
where i.fk_meio_acesso_cd = m.cd
and   i.fk_documento_cd = d.cd
and   i.fk_documento_cd = m.fk_documento_cd
and   b.fk_meio_acesso_cd = m.cd
and   b.cd = i.fk_bloco_servico_cd
and   b.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
order by i.fk_bloco_servico_cd,i.de_bloco_servico,i.dt_item,i.hr_item,
         i.localidade_destino,i.de_servico,i.nr_terminal_destino;

-- OBJETO
Select it.de_bloco_servico, it.dt_item,
       To_char(it.hr_item,'HH24:MI:SS') HrItem,
       it.localidade_destino || '-' || it.estado_destino as LocalidadeDestino,
       it.de_servico, it.nr_terminal_destino,
       it.grupo_horario,
       it.duracao_tarifada duracao, it.vl_liquido as TotLiq,
       it.vl_bruto TotBruto
from ot_documento d, table(d.item) it, table(d.meio) m
where it.fk_meio_acesso_cd = m.cd
and   it.rf_bloco.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
order by it.rf_bloco.cd,it.de_bloco_servico,it.dt_item,it.hr_item,
         it.localidade_destino,it.de_servico,it.nr_terminal_destino;
```

11) Detalhada um documento de um meio de acesso, totalizando por bloco de serviço.

```
-- RELACIONAL
select d.dt_inicio_referencia, d.dt_fim_referencia,
       i.fk_bloco_servico_cd,i.de_bloco_servico,
       sum(i.vl_liquido),sum(i.vl_bruto)
from t_item_faturamento i,t_meio_acesso m,t_documento d
where i.fk_meio_acesso_cd = m.cd
and   i.fk_documento_cd = d.cd
and   i.fk_documento_cd = m.fk_documento_cd
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
group by d.dt_inicio_referencia, d.dt_fim_referencia,
         i.fk_bloco_servico_cd, i.de_bloco_servico;

-- OBJETO
select d.dt_inicio_referencia, d.dt_fim_referencia,
       it.rf_bloco.cd ,it.de_bloco_servico,
       sum(it.vl_liquido),sum(it.vl_bruto)
from ot_documento d, table(d.item) it, table(d.meio) m
where it.fk_meio_acesso_cd = m.cd
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
group by d.dt_inicio_referencia, d.dt_fim_referencia,
         it.rf_bloco.cd, it.de_bloco_servico;
```

12) Detalhamento dos totais gerais de um determinado documento de um meio de acesso.

```
-- RELACIONAL
select d.dt_inicio_referencia, d.dt_fim_referencia,
       sum(i.vl_liquido),sum(i.vl_bruto)
from t_item_faturamento i,t_meio_acesso m,t_documento d, t_bloco_servico b
where i.fk_meio_acesso_cd = m.cd
and   i.fk_documento_cd = d.cd
and   i.fk_documento_cd = m.fk_documento_cd
and   b.fk_meio_acesso_cd = m.cd
and   b.cd = i.fk_bloco_servico_cd
and   b.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
group by d.dt_inicio_referencia, d.dt_fim_referencia;

-- OBJETO
select d.dt_inicio_referencia, d.dt_fim_referencia,
       sum(it.vl_liquido),sum(it.vl_bruto)
from ot_documento d, table(d.item) it, table(d.meio) m
where it.fk_meio_acesso_cd = m.cd
and   it.rf_bloco.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
group by d.dt_inicio_referencia, d.dt_fim_referencia;
```

13) Detalha o total geral de Chamadas de um determinado documento, agrupando pelo período das chamadas.

```
-- RELACIONAL
select d.dt_inicio_referencia, d.dt_fim_referencia,
       sum(i.vl_liquido) TotChamadaLiq,sum(i.vl_bruto) TotChamadaBruto
from t_item_faturamento i,t_meio_acesso m,t_documento d, t_bloco_servico b
where i.fk_meio_acesso_cd = m.cd
and   i.fk_documento_cd = d.cd
and   i.fk_documento_cd = m.fk_documento_cd
and   b.fk_meio_acesso_cd = m.cd
and   b.cd = i.fk_bloco_servico_cd
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
and   b.cd_prioridade_bloco in (10,11,15,19,22,26,28,30,33,37)
group by d.dt_inicio_referencia, d.dt_fim_referencia;

-- OBJETO
select d.dt_inicio_referencia, d.dt_fim_referencia,
sum(it.vl_liquido) TotChamadaLiq,sum(it.vl_bruto) TotChamadaBruto
from ot_documento d, table(d.item) it, table(d.meio) m
where it.fk_meio_acesso_cd = m.cd
and   it.rf_bloco.cd_prioridade_bloco in (10,11,15,19,22,26,28,30,33,37)
and   m.nr_meio_acesso_agregado = '&meio_acesso'
and   d.nr_nota_fiscal = '&NumNF'
group by d.dt_inicio_referencia, d.dt_fim_referencia;
```

14) Recupera o agregador do meio de acesso passado como chave de busca.

```
-- RELACIONAL
select distinct m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,d.vl_consumo,
                m.num_fat, m.nrc, m.fk_documento_cd
from  t_meio_acesso m, t_documento d
where m.fk_documento_cd = d.cd
and   m.num_fat = d.num_fat_agregador
and   m.fk_documento_cd in
      (select distinct d.cd
      from t_documento d, t_meio_acesso m
      where
      d.cd = m.fk_documento_cd
      and  m.nr_meio_acesso_agregado = '&meio_acesso');

-- OBJETO
select distinct m.nr_meio_acesso_agregado AS Agregador,
                d.nr_nota_fiscal,d.vl_consumo,
                m.num_fat, m.nrc, d.cd
from  ot_documento d, table(d.meio) m
where m.num_fat = d.num_fat_agregador
and   d.cd in
      (select distinct d.cd
      from ot_documento d, table(d.meio) m
      where m.nr_meio_acesso_agregado = '&meio_acesso');
```

15) Detalhada os totais do documento agrupando por tipo item da conta.

```
-- RELACIONAL
Select i.de_bloco_servico, i.dt_item, i.de_servico,
       Sum(i.duracao_tarifada) duracaoTotal,
       Sum(i.vl_liquido) as TotLiq, Sum(i.vl_bruto) TotBruto
from   t_item_faturamento i,t_meio_acesso m, t_documento d
where  i.fk_meio_acesso_cd = m.cd
and    i.fk_documento_cd = d.cd
and    i.fk_documento_cd = m.fk_documento_cd
and    i.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and    m.nr_meio_acesso_agregado = '&meio_acesso'
and    d.nr_nota_fiscal = '&NumNF'
group by i.de_bloco_servico, i.dt_item, i.de_servico;

-- OBJETO
Select i.de_bloco_servico, i.dt_item, i.de_servico,
       Sum(i.duracao_tarifada) duracaoTotal,
       Sum(i.vl_liquido) as TotLiq, Sum(i.vl_bruto) TotBruto
from   ot_documento d, table(d.meio) m,  table(d.item)i
where  i.cd_prioridade_bloco in (40,45,30,33,37,10,11,15,19,22,26,28)
and    m.nr_meio_acesso_agregado = '&meio_acesso'
and    d.nr_nota_fiscal = '&NumNF'
and    i.fk_meio_acesso_cd = m.cd
group by i.de_bloco_servico, i.dt_item, i.de_servico;
```

16) Alteração no valor líquido de um item da conta.

```
-- RELACIONAL
update t_item_faturamento i
set i.vl_liquido = vl_liquido + vl_liquido*0.01
where i.fk_documento_cd = &cd_dc;

-- OBJETO
update
    table(select item from ot_documento
          where cd = &cd_dc)
    set VL_LIQUIDO = vl_liquido + vl_liquido*0.01;
```